ALGORITMI

PENTRU

GRAFICA
SI
PROCESAREA
IMAGINILOR

Theo Pavlidis

ALGORITMI PENTRU
PRELUCRARE
GRAFICA SI IMAGINI

CĂRȚI DE INTERES

Ahmad și Fung
Introducere în proiectarea și implementarea computerelor

Andrews
Principii de inginerie firmware în controlul microprogramelor

Arsenault și Roberts
Fiabilitatea și întreținerea sistemelor electronice

Baer
Arhitectura sistemelor informatice

Breuer (ed.)
Automatizare proiectare sisteme digitale: limbi. Simulare și bază de date

Breuer și Friedman
Diagnoza și proiectarea fiabilă a sistemelor digitale

Calingaert
Asamblatori, compilatori și traduceri de programe

Chiar
Algoritmi grafici

Horowitz și Sahni
Fundamentele algoritmilor de calculator

Horowitz și Sahni
Fundamentele structurilor de date

Pohl și Shaw
Natura calculului: o introducere în informatică

Roth
Logica computerizată, testare și verificare

Tomek
Introducere în organizarea calculatoarelor

Ullman
Principiile sistemelor de baze de date

GRAPHICS AND IMAGE PROCESSING

THEO PAVLIDIS
Belt Laboratories

PRESA DE INFORMATICĂ

Copyright ° 1982 Computer Science Press. Inc.

Statele Unite ale Americii.

Toate drepturile rezervate. Nicio parte a acestei cărți nu poate fi reprodusă sub nicio formă, inclusiv fotografii, microfilm și xerografie, și nu în sisteme de stocare și recuperare a informațiilor, fără permisiunea scrisă din partea editorului, cu excepția unui recenzent care poate cita pasaje scurte într-o recenzie sau în conformitate cu Legea privind drepturile de autor din 1976.

Computer Science Press 11 Taft Court
Rockville, MD 20850 SUA

1 2 3 4 5 6 87 86 85 84 83 82

Catalogarea Bibliotecii Congresului în date de publicare
Pavlidis. Theodosios.
Algoritmi pentru procesare grafică și imagini

Include referințe bibliografice și index.
1. Grafică pe computer. 2. Procesarea imaginii.
3. sisteme de recunoaștere a modelelor. 4. algoritmi.
I. Titlu.
T385.P38 001.55 81-9832
ISBN 0-914894-65-X AACR2

Pentru Paul, Karen și Harry

PREFAŢĂ

Evoluțiile tehnologice din ultimii zece ani au făcut populare grafica pe computer și procesarea imaginilor prin computer. Recunoașterea modelelor picturale a arătat, de asemenea, progrese semnificative. În mod clar, există interese care se suprapun între cele trei domenii de cercetare. Afișajele grafice sunt de interes pentru oricine este implicat în procesarea imaginii sau recunoașterea modelelor picturale și multe probleme din grafică necesită metodologii de procesare a imaginii pentru soluțiile lor. Structurile de date utilizate în toate cele trei domenii sunt similare. Se pare că există un corp comun de cunoștințe care stau la baza tuturor celor trei domenii, procesarea *informației picturale de către computer.*

Noutatea acestor domenii face dificilă proiectarea unui curs sau scrierea unei cărți care să acopere conceptele lor de bază. Unele dintre tratatele de grafică se concentrează pe hardware-ul și metodele de interes curent, în timp ce tratatele despre procesarea imaginii pun deseori accentul pe aplicații și procesarea clasică a semnalului. Evoluția rapidă a tehnologiei face ca o astfel de materie să-și piardă relevanța. De exemplu, dezvoltarea fibrelor optice a redus importanța compresiei lățimii de bandă. În mod similar, reducerea prețului dispozitivelor grafice raster a provocat o schimbare majoră a accentului în domeniul graficului. Este important ca un curs introductiv să se concentreze asupra materialului care se așteaptă să fie relevant pe termen lung. Instrumentele matematice folosite în prelucrarea informațiilor picturale par a fi de natură mai permanentă și de aceea am ales să le subliniez. În același sens, am subliniat subiecte în care instrumentele analitice trebuie completate cu considerații euristice sau perceptuale. De exemplu, analiza formei este tratată în doar două secțiuni, chiar dacă subiectul este de mare interes pentru mine și a fost principala mea zonă de cercetare. Pe de altă parte, algoritmii de subțiere au un obiectiv bine definit și pot fi „calul de lucru”* pentru multe scheme de recunoaștere a modelelor. Sper ca cititorul să nu rămână cu impresia că subțierea este în centrul analizei formei. În mod similar, accentul pus pe instrumentele matematice nu implică faptul că acesta este tot ceea ce este necesar pentru proiectarea unui sistem informatic pictural bun. Familiaritate cu inginerie informatică.

fundamentele percepției vizuale umane și o anumită înțelegere a artelor vizuale sunt, de asemenea, esențiale.

Majoritatea materialului acestei cărți a fost folosit de două ori într-un curs pe care l-am predat la Universitatea Princeton, în 1978 și în 1980. Publicul a fost format în

principal din seniori sau juniori în informatică. În plus , unii studenți și studenți absolvenți, precum și studenți în inginerie și științe matematice și fizice, se uită la curs. În general, aveau o experiență solidă în computere și toți erau programatori experimentați. Prin urmare, nu sa făcut niciun efort pentru a introduce concepte simple de informatică. Cursul a fost completat de un laborator, prelegeri susținute de lectori invitați și excursii pe teren, pentru a aduce în atenția studenților stadiul actual al artei. În text, am încercat să mențin aplicațiile și euristicile specifice la minimum.

O provocare specială cu care se confruntă dezvoltatorul unui astfel de curs este dependența procesării informațiilor picturale de un fundal larg matematic și computațional. Sunt necesare calculul, statistica elementară, teoria elementară a graficelor, geometria, procesarea semnalului, structurile dau, analiza algoritmilor și programarea. Este regretabil că specializarea timpurie a studenților face dificilă obținerea unor astfel de baze. Acest lucru nu numai că introduce dificultăți în proiectarea unui curs, dar a reprezentat și un adevărat impediment în progresul cercetării în domeniu. De exemplu, segmentarea imaginii necesită o anumită experiență atât în procesarea stocastică a semnalului, cât și în structurile și algoritmi de date. Curricula de azi oferă rareori șansa unui student de a le dobândi pe ambele.

Textul este organizat astfel: Capitolul I este introducerea care descrie formele datelor picturale. Sunt identificate patru astfel de forme, care corespund aproximativ cu scara de gri (clasa 1). binivel sau binar (clasa 2). curbe (clasa 3). și puncte sau poligoane (clasa 4). Capitolele 2 până la 4 se ocupă de imagini în scala de gri și subliniază transformările și tehnicile statistice. Capitolul 5 este o introducere în tehnicile de reconstrucție utilizate în tomografia computerizată. Un prim curs despre procesarea semnalului este o condiție prealabilă pentru înțelegerea acestor patru capitole, toate tratând imagini de clasa I. Restul textului nu depinde de acest fundal. Capitolul 6 discută structurile de date pentru datele picturale, în timp ce subiectul capitolelor 7 până la 9 este procesarea imaginilor pe două niveluri și subiecte precum trasarea contururilor, umplerea contururilor și subțierea. Toți algoritmii prezentați în aceste trei capitole sunt în esență algoritmi de traversare a graficelor. Se presupune că sunt familiarizate cu terminologia grafică. Capitolele 10 până la 13 sunt despre curbe și potrivire la suprafață. Deși formal depind doar de cunoștințele de calcul simplu, unele
maturitatea matematică este de ajutor. Capitolele 14 până la 17 sunt despre generarea de afișaje grafice, iar metodele de bază sunt cele ale algebrei liniare. Se ocupă în primul rând de imagini de clasa 4.

Unele dintre capitole includ mai mult material decât cel necesar pentru un curs de bază. Acest lucru este valabil mai ales pentru capitolele 8. 9. 10. 11 și 15, unde am inclus algoritmi de bază suplimentari care sunt utili pentru practicanții în domeniu. Pentru un curs de un semestru, cel mai bine este să rămâneți cu un algoritm pentru fiecare dintre umplerea conturului, subțierea sau ping-ul poligonului. În special, se pot selecta cei mai simpli algoritmi. 8.4. 9.4 și 15.2. Pentru un astfel de curs este probabil suficient să acoperiți numai polinoamele Bezier, sau numai spline, sau numai spline cu distribuție uniformă a nodurilor . (O introducere minimă în potrivirea curbei ar putea acoperi numai secțiunile 10.1» 10.2, 11.1 până la 11.3, 11.6 și 11.7 și 10.8, în această

ordine.)

Dacă domeniul de aplicare al cursului este mai limitat decât domeniul de aplicare al textului, atunci unele capitole ar putea fi ignorate cu totul. Un curs cu accent pe grafică ar putea acoperi doar capitolele I, 6 până la 8, 10, 11 și 13 până la 17. Un curs care pune accent pe procesarea imaginilor ar putea acoperi doar capitolele I până la 7 și 9. Capitolele 4, 9 și 12 sunt deosebit de relevante pentru recunoașterea modelelor picturale.

Scrierea acestui text a fost finalizată pe o perioadă de trei ani și jumătate începând cu sfârșitul anului 1977 și s-a desfășurat în trei locuri: Universitatea Princeton, Universitatea din California din Berkeley și, din iunie 1980, la Bell Telephone Laboratories din Murray Hill, New Jersey. O primă schiță a fost finalizată la începutul anului 1980 și a fost folosită ca text pentru curs în timpul semestrului de primăvară de la Princeton. O revizuire extinsă a fost finalizată în timpul iernii anilor 1980-1981.

Am avut norocul să primesc ajutorul multor persoane care au avut amabilitatea să-și lase timpul să citească cu atenție diverse versiuni ale manuscrisului. Doug McIlroy a contribuit cu multe comentarii atât asupra conținutului tehnic, cât și asupra formei manuscrisului. Sugestiile sale au inclus dovezi mai elegante pentru o serie de propoziții și teoreme. Carolyn M. Bjorklund. Lorinda L. Cherry, Chris Van Wyk. și Li-de Wu a făcut multe sugestii constructive asupra întregului text. Părțile care se ocupă de potrivirea curbei au fost îmbunătățite ca urmare a sugestiilor lui Carl de Boor. Larry Shepp mi-a oferit comentarii despre părțile care se ocupă de tehnicile de reconstrucție. Următoarele au ajutat, de asemenea, găsirea erorilor și formularea unui număr de sugestii constructive : Larry S. Davis. Kenneth Fasman, Stein Grinaker. R. Hilbert, Christopher Larson și Murray Loew. A fost o surpriză plăcută să am redactorul Carolyn Ormes să-mi surprindă greșelile nu doar la limbă, ci și la formulele matematice.

Matricele din Figura 3.8 au fost calculate de PC Chen. Cele mai multe din imaginile folosite în exemplele din capitolele I până la 5 mi-au fost oferite de John F. Jarvis și mi-a oferit, de asemenea, facilități de laborator pentru producerea de copii pe hârtie ale afișajelor computerului. Sunt recunoscător pentru originalele oferite de Dr. Stanley S. Siegelman pentru Figura 5.1, Ken Knowlton pentru Figura 6.8, Turner Whitted pentru Figura 17.1 și David M. Weimer pentru Figura 17.2.

Manuscrisul și textul final au fost pregătite folosind diferitele facilități software ale sistemului de operare UNIXt de la Princeton și la Laboratoarele Bell. Textul a fost verificat pentru erori folosind *ortografia, dicția* și *stilul* programelor de Lorinda L. Cherry. Copia finală a fost produsă utilizând programele de formatare *tbl, eqn* și *troff.* Figurile 10.8, 11.4, 11.9, 12.1, 12.7, 12.8 și coperta frontală au fost produse folosind limbajul *pic* de Brian W. Kernighan. (Curbele netede de pe figura de acoperire au fost formate folosind B-spline de tipul descris în Secțiunea 11.7.) Figurile 15.1 și 17.7 au fost produse folosind limbajul *ideal* de Chris van Wyk. Figura 10.1 a fost produsă de o rutină de trasare de uz general care rulează pe aparatul de tipar.

Nu în ultimul rând, vreau să îi mulțumesc soției mele Marion și copiilor mei Paul, Karen și Harry pentru răbdarea și înțelegerea de care au dat dovadă! era preocupat de scrierea cărţii. Foarte des, ajutorul lor a fost mai direct: Marion a introdus în UNIX o

mare parte din text care a fost scris inițial cu mâna lungă, iar Paul a subliniat regulile de umbrire folosite de artiști (Secțiunea 13.10).

Theo Pavlidis

Murray Hill, New Jersey 30 septembrie 1981

t UNIX este o marcă comercială a Bell Laboratories.

TABLE OF CONTENTS

Capitolul 1

INTRODUCERE

1.1 GRAFICA, PROCESAREA IMAGINILOR ȘI RECUNOAȘTERE DE MODEL

Prelucrarea datelor picturale de către computer ia forme diferite într-o varietate de aplicații. A fost obișnuit să se clasifice astfel de lucrări în trei domenii: *grafică, procesarea imaginilor* și *recunoașterea modelelor picturale.*

Grafica se ocupă cu generarea de imagini din informații non-picturale și acoperă diverse aplicații. Complexitatea programelor, precum și efortul de calcul necesar pentru a produce afișaje variază semnificativ, în funcție de sarcină.f Exemplele de afișaje, în ordinea complexității crescânde, includ producerea de diagrame de funcții sau de date experimentale, compoziția de afișaje pentru jocurile pe calculator din ce în ce mai frecvente și producția de scene utilizate în simulatoarele de zbor. Rețineți că, în timp ce intrigile sunt statice în timp, afișajele jocului se schimbă cu timpul, iar scenele simulatorului nu numai că se schimbă în timp, ci trebuie să creeze și iluzia de profunzime. Computer art și animația sunt două aplicații ale graficii care necesită nu numai expertiză tehnică, ci și alte talente. În prezent, acestea par a fi zonele cu cea mai rapidă creștere. Termenul de *grafică interactivă* se referă la dispozitive și sisteme care acceptă informații de la utilizator exprimate în termeni de

t Cititorul nu trebuie să confunde complexitatea programului cu ceea ce se numește adesea complexitate de calcul, efortul de calcul necesar pentru execuția lui. Consultați Secțiunea 1.12 pentru mai multe despre acest subiect.

afișajul pe care îl creează: de exemplu, solicitarea de a trasa o linie între două puncte de pe ecran spre care indică utilizatorul.

procesarea imaginilor se ocupă de probleme în care atât intrarea cât și ieșirea sunt imagini. Sistemele de transmisie a imaginilor, în care cineva se preocupă de

eliminarea zgomotului și compactarea datelor, este un exemplu. Imaginile supraexpuse sau subexpuse sau neclare pot fi îmbunătățite cu tehnici de îmbunătățire a contrastului. Uneori este de dorit să se aplice transformări mai drastice. O imagine cu o gamă largă de iluminare poate fi redusă într-o imagine în care se vede doar două niveluri de iluminare . Siluetele rezultate pot fi reduse și mai mult în figuri de tip stick. Alteori putem chiar să creăm o nouă imagine dintr-un set de altele, cum ar fi construirea de imagini ale secțiunilor transversale ale corpului uman din imagini laterale cu raze X.

Recunoașterea modelelor picturale se ocupă de metode pentru producerea fie a unei descriere a imaginii de intrare, fie a unei atribuiri a imaginii unei anumite clase. Într-un fel, recunoașterea modelelor este problema inversă a graficii pe computer. Începe cu o imagine și o transformă într-o descriere abstractă: un set de numere, un șir de simboluri sau un grafic. Prelucrarea ulterioară a acestor formulare are ca rezultat atribuirea imaginii originale la una dintre mai multe clase. Un sortator automat de corespondență care examinează codul poștal scris pe un plic și identifică cifrele este un exemplu tipic de cerere. Diagnosticul medical automatizat presupune detectarea anumitor anomalii pe radiografii sau alte imagini medicale.

Figura 1.1 rezumă diferențele și asemănările dintre cele trei domenii. Deși fiecare dintre ele are o istorie de cel puțin douăzeci de ani, abia recent au fost subliniate asemănările dintre ele. Legătura dintre recunoașterea modelelor și procesarea imaginii a fost realizată mai întâi deoarece este posibilă transformarea unei imagini în așa fel încât problema clasificării să devină mai ușoară. Realizarea conexiunii dintre aceste două domenii și grafica computerizată este mai recentă. O clasă de probleme de interes comun evident este reprezentarea internă a imaginilor într-un computer: structuri de date, stocare și recuperare, compactare etc. Temeiul comun este mai puțin evident în alte probleme. De exemplu, procesarea imaginii se ocupă adesea de *trasarea contururilor,* în timp ce o problemă la fel de comună în grafică este *umplerea contururilor.* Deoarece o operație este inversa celeilalte, nu este de mirare că anumite întrebări teoretice sunt comune ambelor domenii.

Următorul exemplu descrie o aplicație în care integrarea dintre cele trei zone este completă.

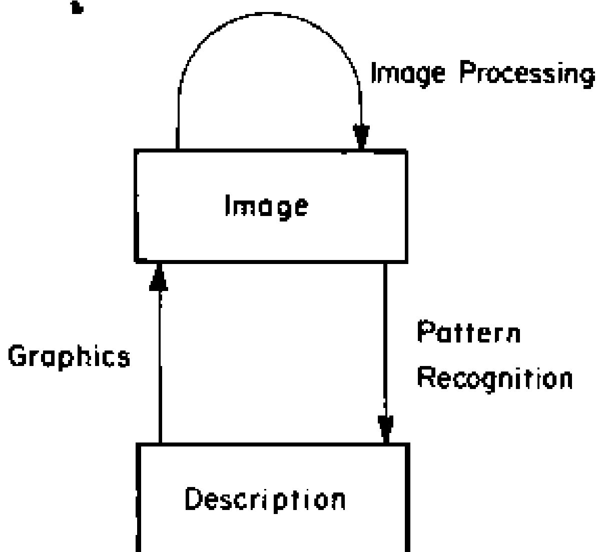


Figura 1.1 Diagrama care ilustrează relația dintre grafică, procesarea imaginilor și recunoașterea modelelor

Exemplul 1.1: O problemă de interes practic este transmiterea imaginilor documentelor prin legături de comunicare. Pentru a avea o reproducere bună a formei unui caracter alfanumeric, trebuie să digitizezi fiecare astfel de simbol într-o matrice de cel puțin 20 de rânduri și 15 coloane (aceasta include o parte din spațiul liber din jur). Reproducerea fidelă a fontului (de exemplu, cursiv, aldine etc.) necesită o rezoluție mult mai mare . Matricea poate fi binară, astfel încât trebuie să trimiteți cel puțin 300 de biți pe caracter. Procesarea imaginii se ocupă de transformări care vor mapa această matrice în mai puțin de 300 de biți, în timp ce aplicarea transformării inverse la receptor va produce imaginea originală, sau cel puțin o bună aproximare a acesteia. Astfel de transformări cunoscute pot reduce numărul de biți necesari cu un factor de cel mult șase. Să presupunem, pe de altă parte, că cineva realizează recunoașterea modelului și pentru fiecare caracter își identifică numele, care este apoi trimis la receptor. Opt biți sunt de obicei suficienți pentru descrierea numelui; prin urmare, cantitatea de date de transmis a fost redusă cu un factor de aproape 40. Desigur, receptorul trebuie să reconstruiască imaginile personajelor, ceea ce este o problemă simplă în grafică. De asemenea, putem sublinia că un caracter poate fi afișat printr-o matrice mai mică: zece rânduri și șapte coloane sunt de obicei suficiente. În acest fel se poate realiza o compactare „permanentă” a datelor. Problema devine mai mare

provocator dacă textul este amestecat cu imagini. Apoi trebuie să comutați între diferite moduri de codare. Referința [l.PC] descrie în detaliu un sistem bazat pe acest principiu. □

Exemplul 1.1 ilustrează un caz în care sunt necesare procesarea imaginii, grafica și recunoașterea modelelor. Astfel de aplicații devin din ce în ce mai numeroase pe măsură ce costul computerelor și al dispozitivelor periferice continuă să scadă. Fiecare dintre aceste trei câmpuri este suficient de larg astfel încât să nu fie posibil să le acopere pe toate trei într-un singur text. În schimb, ne vom concentra pe problemele comune și pe acele întrebări care sunt fundamentale pentru prelucrarea informațiilor picturale. Nu ne vom ocupa de întrebări legate de hardware, nu numai din cauza schimbărilor rapide ale tehnologiei, ci și din cauza accentului pe care îl punem pe algoritmi și metode.

Un aspect al problemei care a fost neglijat în trecut este natura particulară a informațiilor picturale, care necesită metode de analiză diferite decât cele utilizate pentru procesarea semnalelor acustice și electrice. Extinderea procesării semnalului de la una la două dimensiuni este netrivială. Depinde și de interpretarea fizică a dimensiunilor. Astfel, un semnal care depinde de timp și o variabilă de spațiu necesită o procesare diferită decât un semnal care depinde de două variabile de spațiu . Asemănarea matematică dintre forme poate induce în eroare.

1.2 FORME DE DATE PICTORIALE

Este convenabil să distingem patru clase de imagini atunci când vorbim despre procesarea lor de către computer. Clasificarea are mai puțin de-a face cu percepția vizuală reală a imaginilor decât cu modul în care sunt reprezentate și procesate.

1.2.1 Clasa 1: Imagini color și scară completă de gri

Clasa 1 este forma imaginilor obișnuite de televiziune. Astfel de imagini prezintă o reprezentare apropiată a „realității". Imaginile sunt reprezentate ca matrice cu elemente întregi pentru care termenii *element de imagine, pixel* sau *pel,* sunt utilizați în mod obișnuit. În majoritatea aplicațiilor, aceste matrici tind să fie destul de mari: 512X512 este o dimensiune destul de comună. Din acest motiv, ele nu sunt stocate întotdeauna ca simple matrice, iar structurile de date mai elaborate sunt frecvent utilizate. Le vom discuta în Capitolul 6. Imaginile color pot fi reprezentate fie ca trei matrice (pentru roșu, verde și albastru), fie ca o singură matrice în care diferiți biți ai fiecărui element corespund unor culori diferite. De obicei, ochiul uman nu poate distinge niveluri de iluminare care diferă cu mai puțin de un procent, astfel încât

un octet pe culoare pe pixel este suficient. Cu toate acestea, se pot obține rezultate rezonabile folosind trei biți pentru două dintre culori și doi pentru a treia, astfel încât imaginea să poată fi stocată folosind doar un octet per pixel. Vom discuta acest punct mai detaliat în Secțiunea 2.5. Uneori este convenabil din punct de vedere matematic să ne gândim la imaginile color ca la matrice de vectori tridimensionali.

1.2.2 Clasa 2: Imagini pe două niveluri sau „Few Color".

Afișarea unei pagini de text este o imagine tipică de clasă 2 sau cu două niveluri* (alb-negru). Astfel de imagini pot fi reprezentate ca matrici cu un bit per clement, dar și ca „hărți", deoarece conțin regiuni bine definite de culoare uniformă. Acesta este motivul pentru care grupăm imagini de „puține culori" și două niveluri împreună, chiar dacă reprezentarea matricei de biți este bună doar pentru acestea din urmă. O problemă cu utilizarea unui bit per pixel este că nu există o modalitate standard între diferitele computere și dispozitive de afișare pentru a împacheta biți într-un octet și octeți într-un cuvânt. Astfel, pixelul din stânga poate fi împachetat în bitul cel mai puțin semnificativ al unui octet sau în cel mai semnificativ. Prin urmare, utilizatorii trebuie să verifice întotdeauna formularul valabil pentru dispozitivele pe care le folosesc. Vom descrie mai târziu cum pot fi reprezentate hărțile într-un computer. Observați că distincția dintre multe și puține imagini color nu este clară și devine relevantă doar în ceea ce privește forma pe care o folosim pentru a reprezenta imaginile.

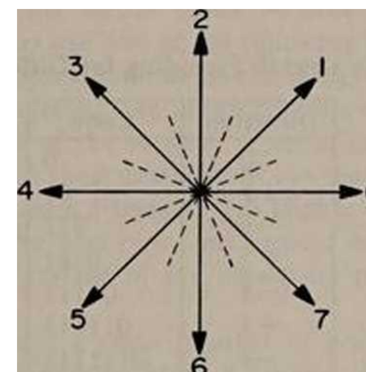


Figura 1.2 Definirea codului de bază al lanțului. Liniile întrerupte delimitează setul de direcții care sunt atribuite unui anumit element al codului.

t Wc preferă termenul „binivel" în locul „binar", deși ambele sunt folosite în literatură. Cuvântul „binar" are o asociere cu computerele și dispozitivele digitale și unii oameni sunt nedumeriți când îl văd folosit pentru semnale analogice.

1.2.3 Clasa 3: Curbe și linii continue

Contururile regiunilor și formele de undă sau diagramele (grafice) sunt exemple de imagini de clasa 3. Datele sunt *secvențe de puncte* care pot fi reprezentate prin coordonatele lor *xy* . Cu toate acestea, această metodă este destul de ineficientă și același lucru este valabil și pentru o reprezentare care utilizează diferențele Ax,Ay în coordonatele dintre punctele succesive. Reprezentări mai eficiente sunt oferite de *codurile în lanț* în care vectorului care unește două puncte succesive i se atribuie un simbol dintr-o mulțime finită. Figura 1.2 prezintă un cod de lanț comun folosind opt direcții . Dacă punctele sunt suficient de apropiate unele de altele, atunci eroarea introdusă de cuantizare poate fi acceptabilă. Dacă punctele au fost alese dintre elementele adiacente ale unei matrice a imaginii (de-a lungul rândurilor, coloanelor sau diagonalelor), atunci reprezentarea lor prin diferențele lor în indicii lor matrici va necesita patru biți pe punct (pentru a reprezenta valorile —1, 0 și +1 pentru fiecare coordonată), în timp ce codul de lanț necesită doar trei biți pe punct.

O formă și mai eficientă este oferită de *codul de lanț diferențial* în care fiecare punct este reprezentat de diferența dintre două coduri absolute succesive. Mai avem opt valori (0, ±1, ±2, ±3 și 4), dar apariția lor nu este la fel de probabilă. Pentru curbele netede ne așteptăm ca valorile 0 și ± 1 să apară mai frecvent decât toate celelalte , în timp ce valoarea 4 va fi extrem de rară. Apoi putem folosi un *cod de lungime variabilă* pentru a reprezenta fiecare direcție. Tabelul 1.1 arată o atribuire posibilă.

Tabelul 1.1: Codificarea cu lungime variabilă pentru codul lanțului diferențial

Direcţie	Cod
0	0
+ 1	01
-1	Ulei
+ 2	0111
-2	01111
+ 3	011111
-3	0111111
4	01111111

O astfel de codificare necesită de obicei nu mai mult de doi biți pe punct în medie. Testele autorului pe o varietate de date au arătat valori medii de 1,8 până la 1,9 biți pentru caracterele alfanumerice și 1,5 până la 1,9 biți pentru contururile obiectelor, cum ar fi un pahar de băut cu pipă sau un șurub. Un cerc mic, care reprezintă un caz nefavorabil, necesita 2,3 biți pe pixel. Dacă o reprezentare a codului în lanț este de dorit pentru curbă

manipularea depinde de aplicație, dar conversia în coordonatele *xy* și invers este simplă (vezi problema 1.4).

1.2.4 Clasa 4: puncte sau poligoane

Imaginile din clasa 4 constau din seturi de puncte discrete care sunt distanțate, astfel încât să nu poată fi reprezentate printr-un cod de lanț. În schimb, trebuie să utilizați o matrice de coordonatele lor *xy* . Punctele pot fi unite prin linii drepte sau alte curbe simple folosind hardware-ul de afișare. Din nou, distincția dintre această clasă și clasa anterioară nu este clară și devine semnificativă doar atunci când luăm în considerare forma reprezentării. Se mai pot folosi coduri în lanț de mai mult de un element pentru punctele îndepărtate, iar alegerea dintre cele două forme de reprezentare trebuie făcută pe baza distribuției statistice a distanței dintre puncte. Într-adevăr, fie / media, iar *L* lungimea maximă dintre puncte. Apoi, un cod de lanț cu lungime variabilă va necesita aproximativ *21* de biți pe punct. O formă *^x,^y* va necesita 21og $_2$ (L). Dacă variațiile de lungime între puncte sunt mici, atunci *L* nu va fi cu mult mai mare decât *I,* iar descrierea prin diferențe de coordonate va fi mai eficientă.

Imaginile de acest tip sunt cele mai des folosite în aplicațiile grafice. Deși afișajul poate fi o imagine de clasa 2, sau chiar o imagine de clasa 1, reprezentarea internă a imaginii este clasa 4. Acest lucru este valabil mai ales atunci când examinăm descrierile obiectelor tridimensionale care sunt proiectate pe planul ecranului pentru a forma imaginea afișată. Majoritatea aplicațiilor folosesc una dintre următoarele forme:

(a) Aproximații poliedrice ale suprafeței. Fețele sunt de obicei triunghiuri. Când este proiectată, imaginea constă din poligoane, (b) aproximări cur viliniare ale suprafeței. Un set de curbe este desenat pe suprafața solidului, iar descrierile lor sunt apoi folosite pentru proiecții , care apar ca imagini de clasa 3, (c) Petice de suprafață de ordin superior. Acestea sunt similare cu primul tip, dar în loc de poligoane plane, elementele care formează suprafața obiectului sunt petice de pe o suprafață de ordin superior.

În toate cazurile, există un număr mic de puncte care specifică poziția obiectului, astfel încât imaginile de clasa 4 sunt într-adevăr de interes central în grafică.

1.3 INTRARE PICTORIALĂ

O imagine analogică trebuie convertită într-o serie de numere înainte de a putea fi procesată de un computer. Procesul se numește *digitizare și* implică alte două procese: *eșantionarea* și *cuantizarea.* Prima se referă la selectarea unui set de puncte peste câmpul de

observare. Caracteristicile imaginii în fiecare astfel de punct sunt măsurate și apoi utilizate pentru o procesare ulterioară. Deoarece toate calculatoarele au memorie finită, trebuie descrise aceste măsurători printr-un număr finit de cifre într-un proces numit cuantizare. Se vorbește adesea despre *rezoluția spațială* pentru a se referi la densitatea punctelor de eșantion, și la rezoluția *de nivel de gri* (sau *culoare)* pentru a se referi la acuratețea reprezentării măsurătorii. Selectarea rezoluțiilor adecvate este discutată în Capitolul 2, în timp ce aici vom oferi doar o scurtă trecere în revistă a hardware-ului utilizat și a instrumentelor software de bază.

O cameră de televiziune este utilizată în multe digitizatoare, deoarece realizează deja transformarea de la semnal optic la semnal electric. Semnalul poate fi apoi eșantionat și cuantificat printr-un convertor *analog-digital* (A/D). Singura problemă este rata uriașă de date care ies de la o cameră standard. Televiziunea comercială transmite 30 de cadre pe secundă, fiecare din aproximativ 500 de linii raster. Dacă vrem să folosim rezoluții verticale și orizontale similare, atunci trebuie să ne ocupăm de aproximativ 30X500X500 de mostre pe secundă. Sunt disponibile convertoare A/D cu capacități de 10 milioane de mostre pe secundă, astfel încât problema nu este atât de tehnologie, cât de cost. De asemenea, fluxul mare de date poate copleși sistemul informatic suport. Dacă procesarea în timp real nu este necesară, atunci se poate modifica camera la o rată mai mică sau se poate folosi un compresor de lățime de bandă. Cel mai simplu mod de a obține această compresie este să eșantionați de-a lungul liniilor verticale. Se așteaptă o scanare raster completă (sau mai multe) înainte de a trece la următoarea linie. Poziția liniei verticale este controlată de computer, care mută linia după golirea tamponului în stocare permanentă. În acest fel se poate obține o reducere a ratei în jur de 500 la 1.

Digitalizatoarele cu scop special folosesc fascicule de lumină controlate cu atenție care scanează imaginea într-un mod predeterminat. Pentru *scanerele cu tambur* imaginea este montată pe un tambur rotativ, în timp ce fasciculul de lumină se mișcă într-o direcție paralelă cu axa tamburului. Astfel de digitizatoare sunt de obicei mai lente decât cele bazate pe camere de televiziune, dar oferă o calitate mult mai bună. Ambele tipuri produc imagini de clasa 1, deși uneori este posibil să se producă imagini de clasa 2 utilizând o cuantizare grosieră . Un alt tip de digitizer este *scanerul de puncte zburătoare* care poate scana imaginea rând cu rând sau poate fi programat să urmărească conturul dintre două zone de intensitate sau culoare diferită, astfel încât rezultatul să fie o imagine de clasa 3. În cele din urmă, *digitizatoarele pentru tablete* oferă o facilitate în care se poate urmări un desen în linie și se poate obține o secvență digitalizată a coordonatelor punctelor trasate. Acest rezultat se obține prin plasarea desenului pe o suprafață magnetizată și identificarea punctelor cu ajutorul unui stylus electric sau al unui cursor . Rezultatul este din nou o imagine de clasa 3.

Digitalizarea directă a obiectelor este necesară pentru anumite aplicații grafice . De obicei, se realizează cu dispozitive speciale cu un indicator care urmărește suprafața unui obiect în timp ce coordonatele acestuia sunt eșantionate. O altă metodă utilizează două proiecții ale obiectului care sunt digitizate simultan folosind un digitizator de tabletă. Apoi, descrierea obiectului tridimensional poate fi reconstruită din cele două desene plane . deoarece un punct din spațiu este complet definit de proiecțiile sale pe două planuri. O zonă promițătoare pentru interacțiunea dintre recunoașterea modelelor/procesarea imaginii și grafică este automatizarea acestor procese. S-ar putea lua diferite vederi ale unui obiect cu o cameră și apoi procesa imaginile rezultate pentru a obține descrierea obiectului. Procesul ar putea fi facilitat dacă s-ar folosi, în loc de imagini cu intensitatea luminii, date *privind intervalul* . Acestea din urmă sunt obținute atunci când un fascicul laser, radio sau ultrasunete este îndreptat către obiect și reflexiile primite în diferite puncte sunt folosite pentru a forma o imagine. Tehnicile de reconstrucție discutate în capitolul 5 ar putea fi, de asemenea, utilizate în acest scop.

Software-ul necesar pentru digitizarea datelor picturale este relativ simplu și nu deosebit de diferit de cel utilizat pentru orice altă digitizare . De obicei, o comandă de tip *citire (dispozitiv, buffer)* va transfera date de la dispozitivul de eșantionare în matricea de buffer. Programul de digitizare conține o buclă cu o astfel de comandă și eșantionarea este încheiată fie atunci când apare un semnal special de întrerupere, fie când au fost colectate un anumit număr de puncte de date (vezi problema 1.1).

Pe de altă parte, este adesea convenabil să aveți sistemul de introducere - încorporat într-un *editor,* un program care permite utilizatorului să modifice datele introduse. De exemplu, se poate dori să selecteze o subimagină a intrării sau să modifice locația unui punct al unei curbe digitizate. Editorii necesită un sistem grafic interactiv iar complexitatea lor variază foarte mult, în funcție de opțiunile dorite de utilizator. Un editor simplu poate fi scris în câteva zile, în timp ce unele dintre pachetele mai elaborate reprezintă mulți ani de muncă. Vom discuta diferite aspecte ale editorilor în părțile corespunzătoare ale textului, începând cu Secțiunea 1.9.

1.4 DISPOZITIVE DE AFIȘARE

În timp ce digitizarea este un proces destul de simplu, conversia datelor în formă picturală este mult mai complexă și este disponibilă o mare varietate de dispozitive. Motivul complexității este că intrarea pe dispozitiv nu trebuie să fie în forma exactă a afișajului, astfel încât dispozitivul trebuie să facă nu numai conversia digitală în analogică (D/A), ci și o anumită cantitate de procesare a datelor. În primul rând, trebuie să se facă distincția între dispozitivele cu *tuburi catodice* și *cele cu raze catodice (CRT)* . Pentru copie hârtie

de obicei se face o singură trecere și, prin urmare, intrarea trebuie sortată pentru a genera comenzile corespunzătoare în ordinea corespunzătoare pentru mișcarea instrumentului de scris. Ecranele CRT nu trebuie să urmeze o ordine strictă de afișare, dar, pe de altă parte, este necesar să repetați procesul, astfel încât imaginea afișată să poată fi vizualizată pentru orice perioadă de timp. *Fotografierii* combină adesea cele două procese în sensul că imaginea este creată pe un CRT și apoi fotografiată pentru a crea copia pe hârtie.

Dispozitivele CRT moderne conțin un microprocesor și un buffer de memorie. Computerul gazdă încarcă informațiile picturale în tampon, apoi microprocesorul scanează tamponul și afișează datele de câte ori este necesar pentru a crea o impresie vizuală. Dacă dispozitivul nu este conectat la un alt computer (adică, dacă „stăpânește singur”), atunci tamponul este umplut prin orice mijloace de intrare sunt disponibile. În ambele cazuri, conținutul tamponului este instrucțiuni pentru microprocesor. Cea mai primitivă formă de instrucțiune este *scrierea (xy a)* care poziționează fasciculul de electroni în punctul (x,y) al ecranului pentru a produce un punct luminos cu intensitate sau culoare proporțională cu *2.* Pentru un afișaj monocrom este nevoie de cel puțin trei convertoare D/A; câte unul pentru fiecare dintre valorile x, y și *z* . Pentru afișajele color, numărul minim de convertoare D/A este de cinci: două pentru coordonate și trei pentru fiecare dintre culori *(2* este acum un vector). Termenul *fișier de afișare* este adesea folosit pentru a se referi la conținutul acestui buffer. Acest fișier este un program de următoarea formă generală:

început:

instrucțiuni de tipul *scrie (x ya)-*

dacă nu a avut loc nicio întrerupere, atunci mergeți la început, sfârșit;

De obicei, microprocesorul efectuează anumite transformări asupra conținutului buffer-ului, astfel încât computerul gazdă (sau utilizatorul în sisteme sund singure) nu trebuie să fie preocupat de detaliile afișajului și să poată pregăti un fișier de afișare cu comenzi de „nivel mai înalt” . Dispozitivele de copiere pe hârtie pot conține, de asemenea, o anumită putere de calcul, dar de obicei este mult mai mică decât cea a CRT-urilor.

Oricât de importantă ar fi diferența dintre CRT și copiere pe hârtie pentru utilizator, aceasta nu este foarte semnificativă în ceea ce privește procesul de conversie sau forma fișierului afișat. Distincția majoră este între grafica *vectorială* și *grafica raster.* (Uneori, grafica vectorială este numită „ grafică calli.) De fapt, unele dintre preprocesarea specială asociată cu dispozitivele de copiere pe hârtie au de-a face cu conversia din vector.

la reprezentări raster și pot fi discutate cel mai bine în acest context. Trecem în revistă aici principalele caracteristici ale fiecărui tip.

1.5 GRAFICA VECTORIALĂ

Dispozitivele grafice vectoriale produc imagini de clasa 3 (sau 4). Instrucțiunile primitive sunt de obicei de următorul tip:

p(xy) poziționați fasciculul de electroni în punctul (xj) *s{z)* setați intensitatea fasciculului la valoarea *z.*

Pentru a afișa un obiect, trebuie să creați o secvență de astfel de instrucțiuni care să-i dea forma. Dispozitivul disponibil poate avea instrucțiuni de nivel superior, fiecare echivalent cu un grup de primitive necesare pentru crearea unor forme comune. Acesta este, de obicei, cazul instrucțiunilor pentru afișarea caracterelor alfanumerice. Acum lăsați Si. S 2 . ■ • 'S„ fie secvențele corespunzătoare fiecăruia dintre *n* obiecte care formează un anumit afișaj Atunci programul executat de procesorul care controlează afișajul este de forma:

început:

Si

$, dacă nu există întreruperi, atunci mergeți la început și sfârșit;

Pentru a modifica aspectul obiectului *i* ,*h* , trebuie să se modifice numai secvența corespunzătoare S r Acesta poate fi cazul, de exemplu, când un obiect se mișcă în fața altora. Nu este nevoie să modificați nicio altă secvență. Există multe modalități de implementare a actualizării; cel mai simplu este să detectezi întreruperile de la procesorul gazdă sau un dispozitiv de intrare.

costum:

Si
Sl

s„
dacă a avut loc o întrerupere, atunci săriți următoarea instrucțiune
mergeți la pornire
proces de intrare
du-te pentru a începe

Sfârşit;

De fiecare dată când se execută bucla, spunem că ecranul este *reîmprospătat.* Rata *de reîmprospătare* este invers proporțională cu lungimea buclei. Dacă

timpul dintre împrospătari este lung în comparație cu degradarea fluorescenței materialului ecranului, atunci se va observa pâlpâirea sau afișajele mai slabe. Prin urmare, există o limită superioară a lungimii buclei (și, prin urmare, a complexității afișajului).

1.6 GRAFICA RASTER

Dispozitivele grafice raster produc imagini de clasa I sau 2, dar pot avea și hardware care permite utilizatorului să simuleze graficele vectoriale . Vom folosi termenul „grafică raster” pentru dispozitivele cu această opțiune și vom numi dispozitivele fără capacitate de afișare vectorială „afișări de imagini”. Caracteristica principală a graficelor raster este o memorie mare cu o locație pentru fiecare locație de ecran adresabilă. Valorile tuturor pixelilor *unei* imagini sunt calculate și stocate în acea memorie, iar apoi afișarea este generată prin citirea memoriei și afișarea valorii fiecărei locații în partea corespunzătoare a ecranului .

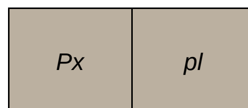
read(I ,x ,y t z) citiți locația de memorie *I* și determinați z din conținutul lui Z. (x și *y* sunt determinate din adresa lui *I.)* Apoi executați o instrucțiune *de scriere (xy^z) .*

Apoi, bucla principală de afișare va fi:

început:

Pentru toate locațiile de memorie / do: begin *read (I ^cy,z) write* (x ^ ,z) end merge to start end;

Dacă televizorul are dimensiunea memoriei de reîmprospătare, atunci controlerul de afișare execută întotdeauna perechi de instrucțiuni TV. Majoritatea dispozitivelor grafice raster folosesc monitoare de televiziune pentru ieșire și este necesar să se execute bucla de cel puțin treizeci de ori pe secundă pentru a produce o imagine fără prea mult pâlpâire. Toate dispozitivele conțin mai mult de un procesor: setul de instrucțiuni care realizează reîmprospătarea afișajului este realizat de hardware dedicat și un alt procesor se ocupă de încărcarea memoriei cu descrierile furnizate de computerul gazdă sau de utilizator. Astfel, lungimea buclei este fixă, iar calitatea afișajului nu depinde de complexitatea acestuia. Cu toate acestea, descrierile obiectelor se amestecă. În grafica vectorială, pentru a elimina un obiect, tot ce trebuie este să eliminați secvența corespunzătoare din bucla de afișare. Lucrurile nu sunt la fel de simple în grafica raster. În special, nu se poate seta intensitatea tuturor pixelilor unui obiect la zero, deoarece acest lucru va șterge și alte obiecte care se suprapun cu acesta. Soluția la această problemă este introducerea *partițiilor* de memorie și afișarea fiecărui obiect pe o partiție separată.



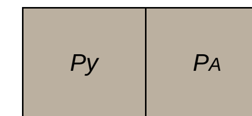


Figure 1.3 Partiția memoriei unui dispozitiv de afișare. Fiecare trimestru se mapează pe același ecran.

Există două implementări posibile. Primul folosește mai mult de o copie a ecranului din memorie. Pentru aranjarea din Figura 1.3, părțile *P }* , *P2* , *P3* și *P4* se adresează toate aceleași coordonate de ecran . Astfel, utilizatorul poate încărca descrierea fiecăruia dintre cele patru obiecte într-o parte diferită. Al doilea folosește *partiția de cuvinte.* Pentru fiecare locație de memorie este rezervat un anumit număr de biți pentru descrierea unui anumit obiect. Dacă conținutul locației ar determina în mod unic luminozitatea și culoarea locației respective a ecranului, atunci o astfel de partiție ar forța culoarea și intensitatea fiecărui obiect. Cu toate acestea, multe dispozitive grafice raster au o caracteristică suplimentară care înlătură această dificultate. Conținutul locației / este folosit pentru a adresa un tabel al cărui conținut determină apoi culoarea și luminozitatea. Conținutul unui astfel de *tabel de căutare video* este definit de utilizator și se poate schimba între afișaje. Astfel, toate obiectele pot folosi aceeași culoare și interval de intensitate.

Exemplul 1.2: Un dispozitiv are 12 biți pe cuvânt, un tabel de căutare și trei convertoare D/A, câte unul pentru fiecare culoare de bază. Cei patru biți inferiori din tabelul de căutare trec în albastru, următorii patru în verde și ultimii patru în roșu. O formă simplă a conținutului tabelului este *identitatea:* dacă valoarea unei locații este *n,* rezultatul tabelului este de asemenea *n.* În acest caz, dacă o locație de memorie conține octal 17, locul respectiv de pe ecran va fi albastru strălucitor. Octal 360 va produce verde strălucitor, octal 7400 roșu aprins. etc. Dorim să afișăm o scenă ca fundal care este disponibil cu trei biți pe culoare plus două obiecte în mișcare, ale căror descrieri sunt disponibile cu doi biți pe culoare. Pentru a crea

display-uri independente trebuie să rezervăm anumiți biți din fiecare cuvânt pentru fundal și obiecte și atunci avem nevoie de 9 + 6 + 6 = 21 de biți, mai mult decât spațiul de stocare disponibil. S-ar putea afișa pe toate împreună, dar apoi, atunci când un obiect se mișcă, va fi necesar să restabiliți imaginea de fundal în memoria de reîmprospătare. Sau am putea sacrifica o parte din gama de culori pentru a obține obiecte și fundal independente după cum urmează: biții 0-5 din fiecare locație sunt rezervați pentru fundal (doi biți pe culoare). Biții 6-8 sunt utilizați pentru primul obiect (unul pe culoare) și biții 9-11 pentru al doilea obiect. Acum, octalul 1000 corespunde celui mai mic bit al celui de-al doilea obiect pe care îl putem afișa pixeli cu acea valoare în albastru. Atunci valoarea respectivă a tabelului de căutare poate fi octalul 10. Octalul 100 corespunde celui mai mic bit al primului obiect și îl mapăm și cu octalul 10. și așa mai departe. Întregul aranjament este prezentat în Figura 1.4. Rețineți că nu ne limităm la corespondența bit cu bit: tabelul de căutare mapează valorile ar fi putut alege, de exemplu, să mapați octalul 100 și octalul 1000 cu octalul 17. etc. Această soluție - presupune că putem aborda selectiv anumiți biți ai fiecărui cuvânt din memoria de reîmprospătare. Acesta este într-adevăr cazul pentru majoritatea dispozitivelor grafice raster (vezi instrucțiunea 13 din Tabelul 1.2). □

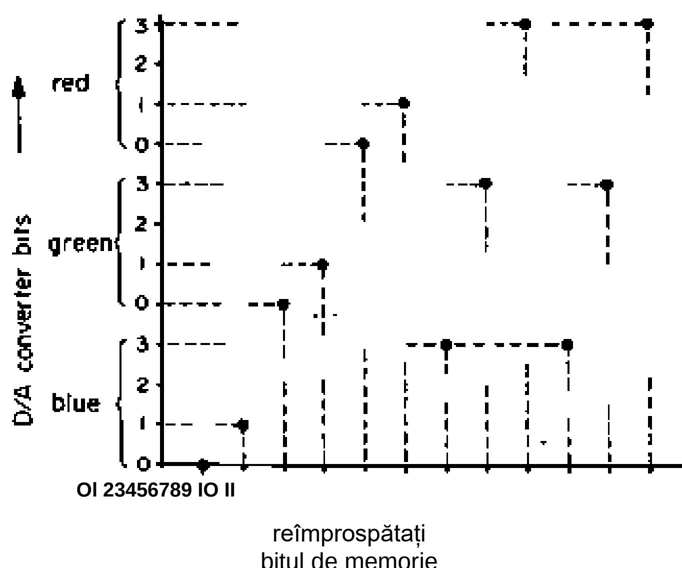


Figure 1.4 Hartă care determină relația dintre culoare și intensitate și biți ai unui cuvânt din memoria de reîmprospătare de exemplu

Tabelul 1.2: Primitive grafice comune

	Nume	Funcţie
1	*seip(x,y)*	Setați punctul curent la coordonate (x.yk
2	*vec(x^)*	Desenați un vector de la punctul curent la un punct cu coordonate (xy), producând astfel o imagine de clasa 3 sau 4.
3	*char {a)*	Afișează simbolul alfanumeric „a” la punctul curent.
4	*câștigă (x,y,X,Y)*	Definiți o zonă de fereastră dreptunghiulară pe dispozitivul de afișare cu colțul din stânga sus la (x,y) și dreapta jos la *(XY).*
5	*înainte {b)*	Setați intensitatea sau culoarea pentru desenarea vectorilor (primul plan) la *b.*
6	*spate (b)*	Setați intensitatea fundalului la *b.*
7	*wpic (buff)*	Afișați conținutul tamponului tampon *în* fereastra definită. Fiecare element al bufferului generează un element imagine, astfel încât este generată o imagine de clasa 1.
8	*wrast (buff)*	La fel ca și instrucțiunea anterioară, dar fiecare element de imagine corespunde unui bit din buffer, astfel încât să fie generată o imagine de clasa 2.
9	*rpic (buff)*	Citiți imaginea afișată pe fereastră în *tamponul tampon.*
10	*şterge*	Setați zona ferestrei la intensitatea fundalului.
1)	*readp (xj)*	Plasați în (xy) coordonatele punctului afișajului indicat de utilizator.
12	*activa (u)*	Examinați cuvântul *u.* și de acum înainte permiteți tuturor celorlalte instrucțiuni să modifice doar acei biți din cuvintele de reîmprospătare a memoriei care corespund biților setați la unu în *u.*
13	*culoare (uv)*	De acum înainte, când un cuvânt are modelul de biți al *lui u,* afișați culoarea determinată de v.

1.7 INSTRUCȚIUNI GRAFICE PRIMITIVE COMUNE

Am menționat mai sus că microprocesorul care controlează un afișaj poate face o anumită cantitate de procesare, astfel încât să nu fie necesar ca utilizatorul să pregătească fișierele de afișare în forma lor cea mai primitivă. De obicei, există un repertoriu larg de instrucțiuni disponibile. Pentru că

Accentul acestui text este pus pe algoritmi mai degrabă decât pe dispozitive, vom folosi termenul *de instrucțiuni primitive* pentru aceștia și vom ignora în continuare orice instrucțiuni de nivel inferior de tipul discutat în cele două secțiuni anterioare. Tabelul 1.2 conține o listă a unor comenzi primitive utilizate în ambele tipuri de dispozitive cu abrevierile care vor fi folosite pentru acestea în acest text. Folosim termenul *punct curent* pentru a desemna ultimul punct adresat pe afișaj. Conceptul este util pentru definirea distanțelor relative.

Argumentele instrucțiunilor *setp* (1), *vec* (2) și *win* (4) pot fi fie *adrese absolute* , fie *adrese relative* în raport cu unele valori prestabilite. Adesea aceste valori sunt coordonatele punctului curent. Forma exactă sau numele unei instrucțiuni depinde de dispozitivul anume, dar cele date în listă sunt tipice. Instrucțiunile *setp* (I) și *vec* (2) există în toate dispozitivele de grafică vectorială și multe dintre ele conțin și instrucțiuni *char* (3) și *fore* (5). Toate sunt comune în grafica raster. Instrucțiunea *readp* (11) este utilizată în grafica interactivă. Acesta permite utilizatorului să puncteze o poziție pe suprafața afișajului, fie cu un stilou luminos, fie cu un cursor controlat de un joystick, trackball sau alt dispozitiv similar.t Ultimele două instrucțiuni sunt utile pentru a stabili partiția de cuvinte în memoria graficelor raster și pentru a permite alocarea de culori și intensități modelelor de biți (reamintim exemplul 1.2).

Afișajele grafice moderne includ adesea alte opțiuni, cum ar fi rotirea unei imagini, operarea pe perechi de imagini etc. Strict vorbind, acestea nu sunt comenzi de afișare, deoarece implică operații în memorie. Deși tendința tehnologică este de a atribui din ce în ce mai multe operații dispozitivelor de afișare, vom presupune aici că numai operațiunile primitive enumerate mai sus sunt disponibile și că operațiunile care implică algoritmi netriviali trebuie să fie furnizate de utilizator. Deoarece acest text se adresează mai degrabă inginerilor decât utilizatorilor de rutină ai echipamentelor grafice, astfel de metode vor fi tratate în detaliu.

1.8 COMPARAȚIA GRAFICILOR VECTORALE ȘI RASTER

Avantajul major al graficii vectoriale este separarea dintre descrierea obiectului și ecran. Prin urmare, rezoluția unui astfel de sistem (adică distanța minimă dintre două puncte de pe ecran) este determinată de electronica hardware-ului care este folosit pentru

t Deoarece wc a decis să nu descrie hardware-ul în detaliu, wc nu descriu astfel de dispozitive aici. Există o varietate de ele pe piață și orice facilitate de grafică are una sau mai multe dintre ele. Este posibil ca dezvoltarea ecranelor sensibile la atingere să facă ca majoritatea acestor dispozitive să fie învechite.

scrie pe ecran. Astfel, matricele de ecran 4096X4096 sunt comune.! În plus, se poate folosi mișcarea fasciculului pentru a desena o linie dreaptă între două puncte într-un mod *analog* și nu digital.

Pe de altă parte, grafica raster necesită corespondență unu-la-unu între pixelii ecranului și locațiile de reîmprospătare a memoriei. Astfel, orice creștere a rezoluției necesită o creștere a dimensiunii memoriei și în prezent, matricea maximă de ecran disponibilă este 1024X1024. Liniile drepte pot fi afișate numai în formă digitală și acest lucru, împreună cu rezoluția limitată, are ca rezultat un aspect asemănător „scării".

Un alt dezavantaj al graficelor raster este că, indiferent dacă memoria este partiționată de-a lungul adreselor sau în cuvinte, numărul de partiții posibile este mic și, prin urmare, numărul de obiecte care pot fi gestionate independent este mic.

Grafica raster afișează imagini de clasa 1 sau clasa 2 fără nicio dificultate. Trebuie doar să plasați valorile corespunzătoare în memoria de reîmprospătare. Este imposibil să afișați imagini de clasa 1 într-un sistem de grafică vectorială fără a le converti efectiv într-un sistem de grafică raster. Imaginile de clasa 2 pot fi afișate cu oarecare efort prin trasarea unor linii paralele strâns distanțate pe zona fiecărei regiuni de o anumită culoare. (Această metodă a fost folosită pentru a produce acest text pe o tipografie foto.)

1.9 EDITOR PICTORIAL

Discuția noastră despre intrările și ieșirile picturale din ultimele două secțiuni s-a ocupat doar de aspectele mai primitive ale acestor operații. În majoritatea situațiilor practice este de dorit să existe mai mult decât un sistem care pur și simplu digitalizează datele picturale. După cum am menționat în Secțiunea 1.3, ar dori să integrăm operațiunile de introducere într-un *editor de imagini.* Folosim acest termen pentru a desemna un program care poate fi folosit pentru a introduce și modifica date picturale în același mod în care un *editor de text* este folosit pentru a introduce și modifica text. *Un astfel de editor oferă, de asemenea, un mediu convenabil pentru implementarea și testarea diverșilor algoritmi pentru procesarea informațiilor picturale. La rândul lor, rezultatele unor astfel de eforturi pot fi adăugate editorului pentru a crește opțiunile pe care le oferă utilizatorului.tt Dintr-un mod practic

t Fotografiatorul folosit pentru a produce acest text este un dispozitiv de grafică vectorială cu un interval de adresă de aproximativ 0-8000.

t Astfel, bucla principală constă dintr-o serie de instrucțiuni „if... tbti (sau un comutator în cazul C. în Pascal, GOTO calculat în FORTRAN etc.)

f Am fi putut alege o altă fracție beudei un sfert, sau am fi permis utilizatorului să aleagă dintr-o listă de rate de reducere sau extindere.

Din punct de vedere de vedere, este probabil necesar să se creeze doi editori, un *editor de imagini* pentru manipularea datelor de clasa I și 2 și un *editor de puncte* pentru manipularea datelor de clasa 3 și 4. Vom discuta despre editorii de puncte la sfârșitul capitolului 10. în timp ce aici vom prezenta editorii de imagini. Părțile esențiale ale unui astfel de sistem sunt un digitizer; un dispozitiv de afișare; și un dispozitiv pentru indicarea pixelilor, de exemplu. unul care implementează instrucțiunea *readp(xy)* din Tabelul 1.2. Cel mai bine este probabil să selectați subimaginile prin mijloace digitale, mai degrabă decât analogice. Astfel, digitizatorul poate scoate datele direct pe memoria de reîmprospătare a dispozitivului de afișare. Apoi utilizatorul selectează o fereastră de pe ecran și folosește o instrucțiune *rpic{buff}* (citește imaginea în buffer) pentru a salva subimaginea corespunzătoare.

Structura de bază a oricărui editor constă dintr-o buclă care este executată în mod repetat. Primul pas al buclei produce de obicei un semn prompt pentru utilizator, care este de așteptat să furnizeze un șir de caractere care descrie operația dorită. Apoi programul examinează șirul și apelează o subrutină adecvată. Uneori, este convenabil să utilizați un *meniu* într-un afișaj grafic pentru a identifica comanda . Utilizatorul indică o zonă a ecranului în loc să tasteze o comandă. Figura 1.5 arată cum ar putea arăta afișajul pentru un editor simplu. Partea din stânga a ecranului este rezervată pentru afișarea imaginilor, în timp ce partea din dreapta listează comenzile din meniu. Presupunem că utilizatorul are mijloacele de a muta două cursoare peste ecran și o modalitate de a semnala sistemului să citească poziția cursorului. (Instrucțiunea *readp(xy}* .) Partea de jos a afișajului listează un set de nume de fișiere.

Următorul mod de operare este tipic. Utilizatorul aduce primul cursor în zona de meniu și semnalează sistemului să execute o instrucțiune *readp(x,y) .* Apoi programul verifică valorile (xj) cu o listă pentru a alege acțiunea necesară. Comenzile „READ TAPE” și „DIGITIZE” determină programul fie să citească un fișier de pe o bandă, fie să colecteze date prin digitizatorul său. În ambele cazuri, imaginea rezultată este afișată pe ecran. Dacă se indică acțiunile „TL CORNER” sau „BR CORNER”. apoi programul execută instrucțiunea *readp(xy}* pentru al doilea cursor, pentru a obține valori pentru coordonatele colțului de jos din stânga sau din dreapta jos. Comanda „SAVE” face ca programul să citească poziția celui de-al doilea cursor pentru a determina un nume de fișier pentru imaginea salvată. Imaginea salvată este cea delimitată de un dreptunghi definit de colțul său din stânga și jos din dreapta. (aproximativ) un sfert din dimensiune, în timp ce „EXPAND”
are efectul opus.t Vom discuta modalități de implementare a ultimelor comenzi, precum și comenzi suplimentare, în capitolele 2 până la 4.

+ +	TL COLT
	BR COLT
	SALVA
	EXTINDE
	MICĂ

				CITEȘTE BANDA
				DIGITIZAȚI
DOSARUL	RLE 2	RLE 3	RLE 4	RLE 5

Figura 1.5 Afișaj utilizat pentru un editor de imagini simplu. Cele două semne încrucișate denotă cursoarele folosite pentru a adresa puncte de pe ecran.

1.10 TRANSFORMĂRI PICTORIALE

Multe dintre problemele discutate în literatură pot fi exprimate ca transformări între clase de imagini și câteva dintre ele ca transformări în cadrul claselor. Le trecem în revistă pe scurt aici.

Clasa I io 2: Acest proces se numește *segmentare* și identifică regiunile în care culoarea și/sau luminozitatea sunt aproximativ uniforme. Termenul se referă adesea la un proces în care se caută uniformitatea în ceea ce privește unele proprietăți mai complexe, cum ar fi textura. Cu toate acestea, în acest text ne vom ocupa doar de schemele de segmentare mai elementare, pe care le vom prezenta în capitolul 4.

Clasa 2 până la 3: O posibilă transformare este *trasarea conturului,* în timp ce alta se *subțiează.* În primul (discutat în Capitolul 7) regiunea este mapată într-o curbă închisă, în timp ce în cea din urmă (discutată în Capitolul 9) este mapată într-un grafic, numit *scheletul* regiunii.

Clasa 3 până la 4: Uneori numit *segmentare curbă,* acest proces încearcă să găsească *puncte critice* de-a lungul conturului. În cazul poligoanelor, aceste puncte sunt colțurile. Astfel de transformări sunt de obicei subiectul literaturii de recunoaștere a modelelor și pot implica metode matematice sofisticate. Unele dintre schemele mai simple pentru găsirea punctelor critice sunt discutate în Capitolul 12.

Clasa 4 până la 3: Aceasta include procesele de *interpolare* (discutate

în capitolele 10 și II), în care o curbă netedă este trasată pentru a trece printr-un set de puncte și *aproximare* (discutată în capitolul 12). în care curba netedă trebuie să treacă în apropierea punctelor.

Clasa 3 la 2: Dacă intrarea este un contur, atunci avem problema *umplerii,* care este adesea exprimată ca cea a *umbririi.* În aceasta din urmă, luminozitatea sau culoarea unei regiuni nu este uniformă, dar variază în conformitate cu unele legi prescrise. Metodele de rezolvare a acestei probleme sunt prezentate în Capitolul 8. Dacă intrarea este un schelet, atunci regiunea trebuie reconstruită prin *expansiune.*

Clasa 2 până la I: Afișarea unei imagini în câteva culori este adesea slabă din punct de vedere estetic, deoarece contururile sunt detectate cu ușurință de ochiul uman (vezi figurile 2.9 și 2.10 de pe planșa 3). O apariție mai netedă poate fi obținută prin utilizarea de *umplere cu trecere joasă* sau prin adăugarea de *zgomot dither.* Aceste tehnici sunt discutate în capitolele 2 și 3.

În linii mari, transformările de la o clasă inferioară la o clasă superioară sunt de

interes pentru recunoașterea modelelor, în timp ce transformările de la o clasă superioară la una inferioară sunt de interes pentru grafică. Procesarea imaginilor se ocupă de ambele, dar și de transformări *în cadrul clasei .* De exemplu, *îmbunătățirea este o transformare în* interiorul clasei, în timp ce *compactarea* este adesea între clasele I și 2.

O clasă importantă de probleme se ocupă cu transformările dintre imaginile bidimensionale și obiectele tridimensionale. Termenul *proiecție* este folosit pentru a desemna o operație care transformă un obiect tridimensional într-o imagine bidimensională sau, într-un caz special, secțiunea transversală bidimensională a unui obiect într-o matrice unidimensională. Termenul *retroproiecție* denotă reconstrucția unui obiect solid (sau a secțiunii sale transversale) din proiecțiile sale. Există două aplicații majore în care aceste probleme sunt importante.

În tomografia axială transversală, o secțiune transversală a unui corp solid este reconstruită dintr-o serie de proiecții cu raze X. Termenul de *algoritmi de reconstrucție* se referă la proceduri pentru rezolvarea acestei probleme. Acestea sunt discutate la sfârșitul capitolului 5.

În grafica computerizată se dorește adesea să se afișeze o anumită vedere a unei descrieri tridimensionale a unui solid. Este ușor să efectuați o transformare geometrică și să găsiți proiecțiile tuturor punctelor obiectului pe planul afișajului. Totuși, o astfel de imagine nu va fi cea a unei vederi, deoarece toate punctele obiectului sunt afișate, chiar dacă în realitate unele dintre ele sunt în spatele altora și, prin urmare, nu sunt vizibile. Astfel, este necesar să existe algoritmi pentru *linie ascunsă* sau *suprafață ascunsă*

îndepărtare. Capitolele 16 și 17 tratează astfel de subiecte. Cele două probleme sunt uneori combinate, așa cum ilustrează următorul exemplu.

Exemplul 1.3: Un medic dorește să aibă opinii diferite despre inima unui pacient. Tomografia axială transversală poate fi utilizată pentru a găsi o secvență de imagini în secțiune transversală care sunt apoi asamblate pentru a forma o descriere a inimii. Un sistem grafic poate fi apoi utilizat pentru a afișa diferite vederi ale orgii. □

Transformările în cadrul claselor de imagini tind să fie relativ simple. Următoarele sunt câteva exemple.

În cadrul clasei 1 sau 2: filtrare. Aceasta include îmbunătățirea contrastului, eliminarea zgomotului de înaltă frecvență etc.

În cadrul clasei 3 sau 4: schimbarea sistemului de coordonate. Aceasta include rotațiile și translațiile.

În cadrul oricărei extinderi de clasă: serie. Transformarea Fourier este cea mai comună dintre ele. Ele sunt adesea folosite pentru compactarea datelor.

1.11 NOTAREA ALGORITMULUI

Algoritmii acestui text sunt dați într-o notație de tip Algol folosind următoarele convenții: Cuvintele cheie sunt afișate cu caractere aldine, variabilele cu *caractere*

cursive și orice altceva în caractere romane. Parantezele sunt folosite pentru a include indici de matrice, precum și argumente de funcție și procedură . Acoladele ({}) sunt folosite pentru a include comentarii. Notația *f (* .1)* înseamnă toate elementele coloanei *I* [1] * a tabloului bidimensional *f.* Implementarea algoritmilor în Algol. PL/I. C, Pascal. Ratfor etc. ar trebui să fie simplu. Implementarea în FORTRAN poate fi mai dificilă, dar sperăm că în anii 1980 majoritatea oamenilor sunt familiarizați cu una dintre celelalte limbi sau știu cum să traducă algoritmi structurați în FORTRAN.

Ilustram notatia noastra oferind ca exemplu un algoritm pentru producerea unei imagini de clasa 1 prin supratiparire. Acesta a fost un instrument de cercetare util înainte de avansarea graficelor raster și este acum un hobby popular. Poate fi, de asemenea, convenabil pentru scopuri de instruire, deoarece imprimantele de linie sunt încă mult mai comune decât afișajele grafice raster. Presupunem că există o matrice *C* care conține codul de supratipărire. Toate literele din coloana *j* ,*k* a matricei sunt supratipărite pentru a codifica nivelul de gri *j.* O alegere bună pentru *C* este dată în Tabelul 1.3. În lista algoritmului *M* indică numărul de rânduri ale lui *C.*

Algoritmul 1.1 Producerea imaginilor în scala de gri prin supratipărire.

1. Pentru fiecare rând /(•) al imaginii faceți: (Nu ne deranjam cu detaliile despre cum este găsit /, deoarece acestea depind de implementarea particulară!

   ÎNCEPE.

2. Pentru *t* de la 1 la *M* faceți:

   ÎNCEPE.

3. Pentru *k* de la I la n face: |/r este numărul de elemente într-un rând de imagini.|

   ÎNCEPE.

4. Setați *p(k)* egal cu *C(ij(k)).*

   Sfârşit.

5. Imprimați matricea *p* fără un avans de linie.

   Sfârşit.

6. Alimentare de linie.

   Sfârşit.

7. Sfârșitul algoritmului.

Instrucțiunea I poate fi implementată fie ca buclă for/do dacă numărul de rânduri este cunoscut, fie printr-o buclă while care verifică o condiție de sfârșit de fișier. Instrucțiunea 2 este simplă, iar instrucțiunea 3 poate fi implementată fie ca buclă for/do, fie ca buclă while dacă sfârșitul rândului este marcat de un caracter nul. În majoritatea limbilor, instrucțiunea 4 va fi *p(J) " C(ij(l)),* deși poate fi necesar să fie executată în doi pași în FORTRAN. În cele din urmă, instrucțiunea 5 necesită cunoașterea caracterelor de control al mașinii. În PL/I comanda SKIP ar trebui să fie omisă, în timp ce în FORTRAN formatat, primul caracter ar trebui să fie * + ' . nu este cazul, atunci trebuie să facem verificări adecvate de indice. Nu vom enumera niciodată astfel de verificări ca parte a algoritmilor în acest text, deoarece presupunem că acestea fac parte din buna practică de programare (vezi de exemplu U.KP] În cazul de față, se poate folosi o

mapare mai sofisticată *între* nivelurile de gri și coloanele de C. astfel încât în pasul 4 ar trebui să scriem o astfel de hartă *(i).* 3.2 O implementare mai eficientă este extinderea *C*- ului original pentru a se potrivi cu numărul de niveluri de gri disponibile. Cu toate acestea, descrierea algoritmilor din acest text nu se va ocupa de astfel de întrebări, deoarece acestea sunt probleme de bună practică de programare.

Persoanele interesate în implementarea algoritmului 1.1 ar putea dori să folosească următorul tabel de valori pentru matricea *C.* Acesta a fost propus de Henderson și Tanimoto ll.HT]. Hârtia lor conține unele
informații interesante pentru cei suficient de motivați pentru a încerca să implementeze algoritmul în limbaj de asamblare.

Tabelul 1.3: Matricea de caractere pentru producerea scalei de gri prin supratipărire

M)MM®NHI<HHHXHXOZ»W}lO<>S . | •*•■:-.-•
.. .. - • - -
######O).-
ooooo
00*

1.12 CATEVA CUVINTE DESPRE COMPLEXITATE

Orice discuție despre algoritmi trebuie să acopere subiectul complexității computaționale și, ori de câte ori prezentăm un algoritm în acest text, vom face un efort pentru a da o idee despre timpul și spațiul necesar pentru execuția lui. Aici am dori să subliniem o eroare comună pe care o fac mulți studenți: complexitatea de calcul și de programare confuză. În general, lungimea programului care implementează un algoritm are puțin de-a face cu viteza de execuție sau chiar cu cerințele de memorie. Dacă există vreo relație, aceasta tinde să fie în direcția opusă. Algoritmii „complicați” sunt, de obicei, mai rapidi decât algoritmii „simpli“.

Este adesea tentant să folosiți forma recursivă a algoritmului deoarece este mult mai scurtă decât cea nerecursivă și numărul de operații al celor două forme este același. În astfel de cazuri, trebuie să aveți în vedere costul ridicat al apelurilor recursive, necesitatea ca mașina să salveze valorile din registru etc. Dacă numărul de astfel de apeluri este mic în comparație cu numărul celorlalte operațiuni, atunci acest cost poate merita simplitatea rezultată în program. În caz contrar, forma nerecursivă dă programe mai eficiente.

În timp ce simplitatea programării poate părea atractivă pentru un student care trebuie să respecte un termen limită și intenționează să își ruleze programul pe date limitate, este în detrimentul oricărui mediu de aplicații în care un program este rulat pe volume mari de date.

1.13 NOTE BIBLIOGRAFICE

O serie de texte despre procesarea imaginilor și grafică discută despre designul hardware al afișajelor, digitizatoarelor etc. I1.NS) și (I.BO) sunt locuri bune pentru a începe. Stadiul tehnicii în acest domeniu se schimbă rapid și este recomandabil să urmăriți cu atenție ceea ce oferă producătorii. Publicațiile societăților și întâlnirilor profesionale sunt o sursă bună. În Statele Unite ale Americii, ACM Special Interest Group on Graphics (SIGGRAPH) publică trimestrial *Computer Graphics* și ține o întâlnire anuală (de obicei în timpul verii) care include un târg important de echipamente. Detalii despre această întâlnire se găsesc consultând calendarul Comunicărilor *ACM.* Publicațiile IEEE *Spectrum. IEEE Computer Graphics* și *Computer* conțin întotdeauna secțiuni cu reclame și anunțuri pentru hardware și acestea includ adesea echipamente de procesare a imaginilor și grafică.

Lucrări de cercetare în domeniul general al prelucrării informației picturale apar și în următoarele reviste. *Computer Graphics and Image Processing* (abreviat *CGIP), IEEE Transactions and Pattern Analysis and Machine Intelligence,* noile anunțate *ACM Tranzacții pe grafică* etc.

1.14 LITERATURA RELEVANTA

[I.[80] Booth, KS *Tutorial: Grafică pe computer.* New York: IEEE, 1979. [l.HT] Henderson. P. şi Tanimoto. S. „Considerații pentru o ieșire eficientă a imaginii prin intermediul imprimantei de linie.” *CGIP.* 4 (1974). p. 327-335.

II.KP| Kernighan, BW și Plauger, PJ *Software Tools.* Reading, Mass: Addison-Wesley. 1976.

[I.NS] Newman. WF și Sproull, *Principiile RF ale comunicațiilor interactive Puter Graphics.* 2 W ed. New York: McGraw-Hill, 1979.

[I.PC] Pratt. WK: Capitant. PJ; Chen. WH; Hamilton. ER; și Wallis. RH „Sistem de comprimare a datelor cu potrivire combinată a simbolurilor fax”. *IEEE Proceedings. 68* (1980). p. 786-796

1.15 PROBLEME

1.1. Scrieți un program de digitizare pentru unul dintre dispozitivele disponibile în instalația dvs.

1.2. *Împachetarea biților:* o imagine în scară de gri conține doar două valori distincte ale nivelurilor de gri, așa că poate fi codificată ca imagine pe două niveluri. Să presupunem că toți pixelii au acum valoarea 0 sau I. Scrieți un program pentru a stoca imaginea cu opt pixeli pe octet.

1.3. *Despachetarea bitului:* Pentru a afișa imaginea anterioară trebuie să

au un octet pe pixel Scrieți un program pentru a-l „despacheta”.

1.4. *Cod în lanț:* Scrieți un program pentru găsirea coordonatelor *xy* ale unui pixel a cărui direcție a codului în lanț din alt pixel este cunoscută. De asemenea, implementați operația inversă, găsind codul lanțului din coordonatele .xy.

1.5. Implementați editorul simplu descris în Secțiunea 1.9, cu excepția comenzilor EXPAND și SHRINK. Ar trebui să încercați să faceți programul cât mai

independent posibil de dispozitiv. De exemplu, puteți crea o procedură care preia coordonatele cursorului și identifică comanda din acestea. În loc să apeleze procedura pentru a executa acțiunea corespunzătoare acolo, procedura de intrare ar trebui să returneze programului principal un șir cu numele comenzii. În acest fel, editorul principal nu are nevoie să cunoască detaliile comenzilor grafice.

1.6. *Scară de gri prin supratipărire:* Implementați algoritmul 1.1. Dacă doriți, puteți utiliza codul din Tabelul 1.3.

Capitolul 2

DIGITIZAREA IMAGINI ÎN SCANĂ DE GRI

2.1 INTRODUCERE

Când o imagine urmează să fie procesată de computer, este adesea descrisă ca o matrice sau o altă structură de date discretă. Dar o imagine este în primul rând un semnal care transmite informații unui observator și există multe aplicații în care acest aspect este deosebit de important. Vom dedica acest capitol și următorul capitol discutării unor astfel de probleme, în special pentru imaginile în scala de gri (clasa 1). Prima problemă este conversia unei imagini continue într-o formă discretă, iar aceasta implică două procese: *eșantionarea,* care este selectarea unei grile discrete pentru a reprezenta o imagine, și *cuantizarea,* care este maparea valorilor de luminozitate și culoare în numere întregi. În grafică se preocupă probleme similare: în special alegerea rezoluției afișajului și numărul de niveluri de gri sau culori. Aceste procese sunt relevante și în datele unidimensionale și au fost studiate amănunțit în acest caz, dar datele bidimensionale prezintă noi probleme. Vom dedica prima parte a acestui capitol unei revizuiri a tehnicilor de transformare și apoi vom discuta eșantionarea pentru cazul unidimensional, urmată de eșantionarea pentru imagini. Cuantizarea va fi tratată în ultima secțiune.

2.2 O REVIZIE A FOURIER ȘI A ALTE TRANSFORMĂRI

Transformele Fourier sunt un instrument comun în procesarea semnalului și sunt definite atât pentru funcții unidimensionale, cât și bidimensionale. Următoarele două

formule sunt valabile pentru cazul continuu: Dacă /(r) este o funcţie definită pe un interval lO.co). atunci transformata sa Fourier este

co

H*) ■ *ff(t)**?(-}**)dt* 0<w<co (2.1) z-0

Dacă/(x^) este o funcție a două variabile definite pe planul infinit, atunci transformarea bidimensională a lui /(xj) este: co co

F(u,v)~f J *f{xy)cxp(-j(xu+yv))dxdy* 0<u,v<co (2.2) - 03 ” CO

Termenul *spectru* este adesea folosit ca sinonim pentru transformata Fourier. Transformarea poate fi definită și pentru funcțiile al căror argument este un număr întreg de la 0 la /V—]. Astfel de funcții pot fi considerate ca fiind constând din *N* eșantioane dintr-un semnal continuu, dar acest lucru nu trebuie să fie întotdeauna cazul. În mod formal, avem pentru cazul unidimensional

*■(“)= S /(*)exp(-j-“—u4)   0<u<A'-l   (2.3)

iar pentru cele bidimensionale

wi wi   -

F(u,v)-2 2 /(*./)exp(-;-^-(*u+/v)) 0<w,v<^-l (2,4) *-0 /-0   ^

Ecuațiile de mai sus sunt de obicei denumite *transformată Fourier discretă (DFT).* Este posibil să se facă diferite limite ale celor două sume și să se definească *DFT* pe un domeniu dreptunghiular. Cu toate acestea, nu există nicio modalitate de a defini transformarea Fourier bidimensională pe o regiune de formă arbitrară. Dacă restrângem limitele integrării în ecuația (2.2). sau limitele de însumare din ecuația (2.4), atunci rezultatul va depinde nu numai de valorile lui/, ci și de forma domeniului de însumare/integrare f *Transformata Fourier inversă discretă {DIFF}* este definită ca

f< *k* >~ ^^ F(u > CX PO ^ uk)   0<A<NI (2,5)

t Se poate arăta că (integrala rezultată este egală cu convoluția complexă a transformării Fourier a lui / cu transformata Fourier a funcției caracteristice domeniului de integrare.

şi

/ *Nl N*-\ *j*

f^' 1 ^^^ S ^-^expU““^^/)) 0^,/</Vl (2.6)

Ecuațiile de transformare pot fi scrise în multe moduri, dintre care unele sunt utile în motivarea altor transformări, precum și modalități eficiente de calcul. Să notăm exponențiala exp(—y-^-) cu *2* și să definim matricea *Z* după cum urmează:

eu

$$Z = \begin{vmatrix} 1 & 1 & 1 \\ 1 & z & z^2 \\ 1 & z^2 & z^4 \end{vmatrix}$$

pw-D (2.7)

zCV-DIN-l)

1 z*“1 z2^-1’

Cu alte cuvinte, Z^ ”2”*. Pentru majoritatea aplicațiilor *N* este nu numai par, ci și o putere de 2. Matricea *Z* poate fi simplificată observând că 2* - 1, *z* Nn - -1, *z* N/ * - *-j.* și *z* WH *= j.* De asemenea, observăm că este o matrice simetrică și că produsul scalar al oricărei coloane (sau rânduri) cu o altă coloană (sau rând) este zero. Produsul scalar al unei coloane (sau rând) cu ea însăși este *N.* Astfel, *Z* are proprietatea importantă că inversul său este egal cu transpunerea conjugată ori I//V. O matrice precum 1/V^VZ, unde inversul este egal cu transpunerea conjugată , se spune că este o matrice *unitară* . Fie acum *f* un vector coloană ale cărui elemente sunt valorile *f(k)* și *F* un vector coloană cu elementele F *(u).* Atunci ecuația (2.3) poate fi scrisă sub formă de matrice ca

F-Zf . (2,8)

Pentru cazul bidimensional observăm că transformarea este echivalentă cu înmulțirea mai întâi a tuturor coloanelor *off(k,l)* cu Z, iar apoi a tuturor rândurilor sale cu *Z'* (transpunerea lui Z) din dreapta. Atunci ecuația (2.4) devine

F - ZfZ'. (2,9)

Expresiile / și *F* denotă matricea de imagine și matricea de transformare. Expresii similare pot fi găsite pentru transformarea inversă, dacă *Z* este înlocuit cu *\/N* ori transpunerea sa complexă conjugată *Z .*

Acum se pot defini transformări suplimentare utilizând ecuațiile (2.8) și (2.9) și schimbând definiția lui *Z* atâta timp cât *M^NZ* este o matrice unitară. O astfel de alegere dă *transformarea Hadamard.* Matricele pentru acea transformare pot fi definite recursiv după cum urmează:

$$\begin{matrix} 1 \\ -1 \end{matrix} \Bigg] \qquad (2.10a)$$

$$Z^{\wedge\wedge} \quad \begin{matrix} Z_N & Z_„ \\ Z_„ & -z_N \end{matrix} \qquad (2.10b)$$

Astfel, transformata Hadamard corespunde expansiunilor de-a lungul undelor pătrate, mai degrabă decât undelor sinus și cosinus ale transformării Fourier. În plus, matricea Z ar putea fi pusă într-o formă similară cu ecuația (2.7) cu z = —1, dar cu o regulă diferită pentru modificarea exponenților. Avantajul major al transformării Hadamard este că înmulțirile complexe sunt înlocuite cu schimbări de semn, un aspect important în primii ani de calcul, unde unitățile aritmetice în virgulă mobilă erau greu de găsit. În caz contrar, produce rezultate similare cu cele obținute prin transformarea Fourier.

Aplicația majoră a transformărilor este în proiectarea filtrelor pentru îmbunătățirea imaginii și, de asemenea, în anumite tehnici de compactare a datelor. Ele oferă, de asemenea, bazele pentru reconstrucția imaginii din proiecții, tehnici pe care le discutăm în Capitolul 5. Utilizarea lor pentru compactarea datelor este justificată în cazurile în care majoritatea F(u) sunt zero. Atunci ecuația (2.5) sau (2.6) poate fi utilizată pentru recuperarea valorilor lui *f {k)* din mai puține de *N* valori ale transformării. Cu toate acestea, utilizatorul ar trebui să fie conștient de faptul că fiecare dintre elementele de transformare poate necesita mai mulți biți pentru descrierea sa decât valorile funcției originale. Acolo deci. raportul de compactare poate să nu fie atât de mare pe cât s-ar putea aștepta.

Utilizarea ecuațiilor definitorii pentru calculul unei transformări date de ecuația (2.8) necesită N^2 calcule pentru cazul unidimensional și /V ^{4} pentru cazul bidimensional. Este posibil să rearanjați termenii sumei din ecuația (2.3) și să folosiți faptul că produsul *uk* ia aceleași valori pentru diferite valori ale *lui u* și *k* pentru a ajunge la un algoritm mai eficient, *transformata Fourier rapidă (FFT).* Acest lucru este descris în Anexa 2.A, unde se arată că numai operațiile Wlogj/V sunt necesare pentru cazul unidimensional. Algoritmul rapid de transformare Fourier poate fi folosit pentru a evalua transformarea bidimensională, găsind mai întâi transformarea fiecărui rând al imaginii și apoi transformarea tuturor coloanelor. Acest lucru este prezentat în algoritmul 2.1, unde *FFT(N ,x)* este o procedură care înlocuiește tabloul *N* elemente *x* cu transformarea sa Fourier discretă.

Algoritmul 2.1 FFT bidimensional

1. Pentru /-0 lo /V—eu fac:
   ÎNCEPE.
2. Copiați/(•,/) în tabloul x.
3. Apelați *FFT(N,x).*
4. Înlocuiți /{*,/) cu x.
   Sfârşit.
5. Pentru A: = 0 la/V-1 faceți:

ÎNCEPE.
6. Copiați/(A,*) în tabloul x.
7. Apelați FF7(N,x).
8. Înlocuiește /(*,•) cu x.
Sfârşit.
9. Sfârșitul algoritmului.

Fiecare apel la *FFT(N,x)* implică un cost de calcul proporțional cu NlogjN. Deoarece există astfel de apeluri de 27V, costul total va fi de ordinul $/V^2 \log_2 (N^2)$.

2.3 PRELEVARE

Pentru a procesa o imagine sau orice alt semnal de către computer, trebuie să o transformăm într-un set finit de numere. Eșantionarea este selecția unui set de puncte discrete în continuum-ul de timp sau spațiu. Doar valorile semnalului din acele puncte vor fi utilizate în procesarea ulterioară.

2.3.1 Eșantionarea unidimensională

În cazul unidimensional rezultatul matematic fundamental este teorema de eșantionare a lui Shannon, care este exprimată în termeni de spectru al semnalului. Fie *d(t)* reprezentarea digitală a/(r), constând din impulsuri necuantificate la intervale de *T* unități, astfel încât

d(kT)~f(kT) *-0,l,'--co (2.11a)

d(r)“O ifr^AT (2.11b)

Atunci se poate demonstra că transformata Fourier D(w) a lui 4(r) este legată de F(w) prin ecuație

D(w) = 2 (2,12)

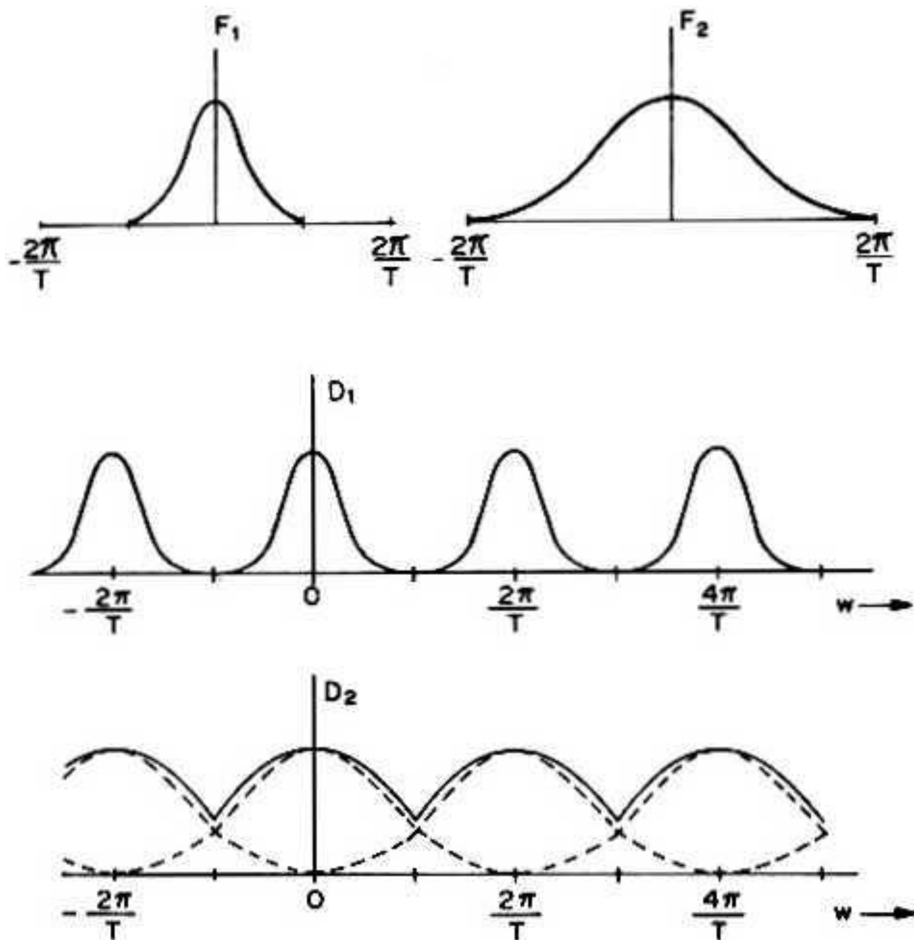


Figura 2.1 Transformată Fourier a semnalelor eșantionate. Rândul de sus arată spectrele F ((w) și F^w) a două semnale neeșantionate: /i(r) și *f^i).* Diagrama din mijloc arată spectrul D ((w) al semnalului eșantionat d ((r). în timp ce diagrama de jos arată ulei*). FJw) este zero în afara unei benzi de frecvență, dar acesta nu este casc cu F^w). Liniile întrerupte din diagrama de jos arată forma transformării semnalului continuu repetat la fiecare *2τ/T* unități de frecvență.

Cu alte cuvinte, transformata Fourier a lui *d(t)* constă dintr-o sumă a deplasărilor transformatei Fourier a lui *f(i).* Rețineți că ecuația (2.12) nu se referă la transformata Fourier discretă a lui *d(t),* ci la transformarea continuă a unei funcții care se întâmplă să fie diferită de zero numai în anumite momente discrete. Diferența dintre transformările funcțiilor *d* și / poate fi văzută din exemplele din Figura 2.1. În diagrama din stânga sus.

F(w) este zero pentru *w>r/T,* în timp ce acesta nu este cazul în diagrama din dreapta sus. Graficele lui D(w) pentru aceste două cazuri sunt prezentate în diagramele din mijloc și, respectiv, de jos. În primul caz, dat D(w). putem găsi *F(w)* exact și, prin urmare, putem reconstrui *f (i)* din *d(t).* Acest lucru nu este posibil în al doilea caz din cauza suprapunerii termenilor din ecuația (2.12). Aceasta conduce la următoarea formulare a teoremei de eșantionare:

Teorema 2.1: *(Teorema de eșantionare a lui Shannon)* Fie w^ cea mai mare valoare a lui *w* pentru care *F(w)* este diferit de zero. Atunci *f (t)* poate fi reconstruit exact din probe dacă timpul dintre probe este mai mic decât

"amesteca

adică, frecvența de eșantionare este de cel puțin două ori dimensiunea lui w_{mi} .

Rețineți că această teoremă nu sugerează o modalitate de reconstrucție a semnalului continuu din eșantioanele sale discrete. Spune doar că se poate. De fapt, este necesar să se utilizeze tehnici destul de sofisticate pentru a reconstrui un semnal atunci când este eșantionat la frecvența minimă. Dacă cineva este limitat la o anumită formă de reconstrucție, atunci frecvența de eșantionare ar putea trebui să fie mult peste minimul sugerat de teorema de eșantionare.

Exemplul 2.1: Să presupunem că reconstrucția constă dintr-o constantă, egală cu valoarea eșantionului, așa cum se arată în Figura 2.2. Fie /(/) - Esin(wf) pentru unele w. Dacă dorim o eroare maximă între semnalul reconstruit și original de cel mult o cantitate dată e. atunci Figura 2.2 arată că următoarea ecuație ar trebui să fie valabilă

l/(*r±y)-/(*r)l <^ (2,13)

unde * este un moment de eșantionare și *T* este intervalul de eșantionare. Teorema valorii medii a calculului poate fi apoi utilizată pentru a arăta că această calitate ine va fi valabilă dacă valoarea maximă a valorii absolute a derivatei lui /(O. *f'(t).* este mai mică de *2e/T.* Pentru alegerea particulară a lui/(r) putem calcula, de asemenea, maximul diferenței din ecuația (2.13) printr-o formulă trigonometrică și să găsim că această ecuație este echivalentă.

£lsin(-yOl <« •

Dacă aproximăm sinusul unui unghi mic cu unghiul însuși, constatăm că inegalitatea de mai sus devine

T *eh)* (214)

dând o limită mult sub limita Shannon. Același răspuns se găsește dacă folosim teorema valorii medii. □

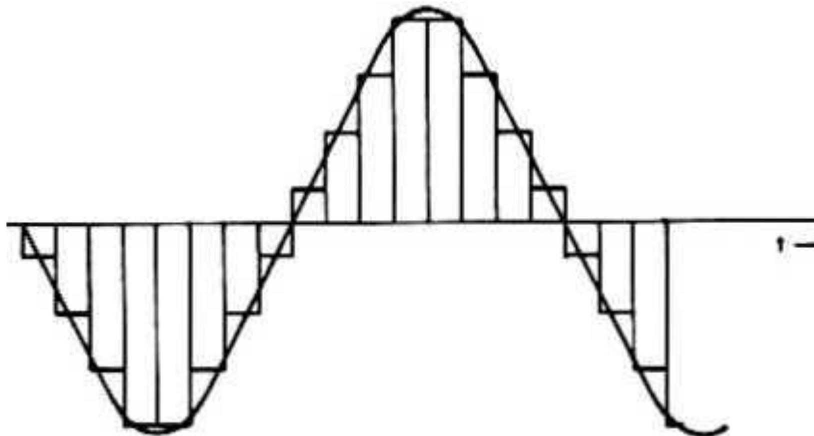


Figura 2.2 Reconstrucția semnalului printr-o aproximare constantă pe bucăți

2.3.2 Eșantionarea bidimensională

Alegerea formei de reconstrucție este de obicei limitată în procesarea imaginilor și acest fapt trebuie reținut ori de câte ori se folosește o generalizare a teoremei de eșantionare în două dimensiuni. Chiar dacă teorema lui Shannon poate fi extinsă trivial la două dimensiuni, nu este deosebit de relevantă din cauza limitărilor algoritmilor de reconstrucție disponibili. Prin urmare, datele picturale trebuie de obicei eșantionate la o rată mult mai mare decât ceea ce ne-am putea aștepta de la o analiză spec trală. În special, se poate folosi o limită superioară a formei ecuației (2.14). Intervalul de eșantionare rezultat va fi *de 2e/(Er)* ori valoarea cerută de teorema de eșantionare. Dacă *e/E* este de 1%, atunci trebuie să eșantionăm aproximativ 157 de tei la fel de des decât ceea ce este cerut de limita Shannon!

Figurile 2.3 până la 2.5 (Plăcile 1 și 2) ilustrează problema. Primul

este o imagine la o rezoluție înaltă (256x256) care pentru majoritatea observatorilor nu se poate distinge de o imagine analogică. Celelalte două au fost obținute din primele prin sărirea eșantioanelor, astfel încât Figura 2.4 constă din 64x64 eșantioane și Figura 2.5 din 32x32. Acestea sunt afișate pe un ecran mai mare prin repetarea valorii fiecărei probe de 16 și, respectiv, de 64 de ori. Sunt cu siguranță de calitate scăzută, dar acest lucru nu se datorează doar subeșantionării. Cititorul poate încerca să le privească de la distanță sau strâmbându-și ochii. Calitatea imaginilor se îmbunătățește deoarece majoritatea informațiilor sunt acolo. Reconstrucția constantă pe bucăți este cea care introduce distorsiunile aparente. Este posibil să se dezvolte algoritmi care interpolează liniar între eșantioane (vezi problema 2.3), astfel încât, dacă o imagine cu rezoluție mică este afișată pe un ecran cu rezoluție înaltă, pixelii suplimentari să aibă valori intermediare între cele ale mostrelor date. Cu toate acestea, procesul de interpolare poate fi destul de lent, astfel încât utilizarea sa cu un afișaj convențional nu este practică. Numai atunci când sunt dezvoltate afișări grafice care vor face această interpolare local (prin hardware special) va fi fezabilă eșantionarea imaginilor mai aproape de rata prezisă de teorema

lui Shannon. (În terminologia filtrului digital, aceste dispozitive vor fi echivalente cu elemente *de reținere de ordin înalt* .)

Necesitatea supraeșantionării este adesea un obstacol în calea dezvoltării unui sistem digital complet de procesare a informațiilor picturale. Un posibil compromis este combinarea ieșirii analogice și digitale prin utilizarea tehnologiei de televiziune. Un monitor primește semnale de la două surse: un controler grafic raster și o cameră de televiziune sau o bandă video analogică. Este posibil să amestecați cele două și să creați o suprapunere de. cuvânt. informaţii statistice (produse digital) peste harta (analogică) a unui oraş.t

Definirea punctelor de eșantionare este simplă pentru semnalele unidimensionale, dar probleme serioase apar în două dimensiuni. Definim formal unii dintre termenii pe care îi vom folosi frecvent .

Definiția 2.1: Fie *P* planul care conține o imagine analogică. O *celulă de digitizare este* definită ca o submulțime compactă și convexă a lui *P* peste care se calculează valoarea unui eșantion din imaginea digitizată. Unirea tuturor acestor celule este definită ca *grila (de eșantionare).* Un singur eșantion se numește *element de imagine,* sau *pixel,* sau *pel.* Dacă *D* este planul de afișare, atunci pixelii sunt mapați pe *celulele de afișare* care au ca rezultat imaginea analogică reconstruită . □

t Acest aranjament a fost descris de Dr. Jean-Paul Jacob de la IBM Research - Laboratories.

Ar trebui să subliniem că nu presupunem că celulele de digitizare sunt disjunctive. În majoritatea dispozitivelor de digitizare, acestea se suprapun, deși celulele de afișare nu se suprapun de obicei. Figura 2.6 prezintă o grilă tipică de eșantionare și afișajul corespunzător. Deși teoretic este posibil, și uneori practic inevitabil, să se utilizeze diferite grile pentru eșantionare și afișare, acest lucru nu este recomandabil din cauza distorsiunilor rezultate. Majoritatea procedurilor de digitizare sunt caracterizate de o *funcție de distribuire a punctelor. g(r),* care reflectă modul în care diferitele puncte - contribuie la valoarea unui eșantion în funcție de distanța lor *r* față de acesta. Funcția *g(r)* este o funcție descrescătoare a *lui r* și este zero în afara celulei de digitizare. Dacă *g(r)* scade suficient de repede, atunci se pot presupune celule care nu se suprapun și, ca primă aproximare, folosiți termenul „celulă” pentru a se referi atât la digitizare, cât și la afișarea celulelor.

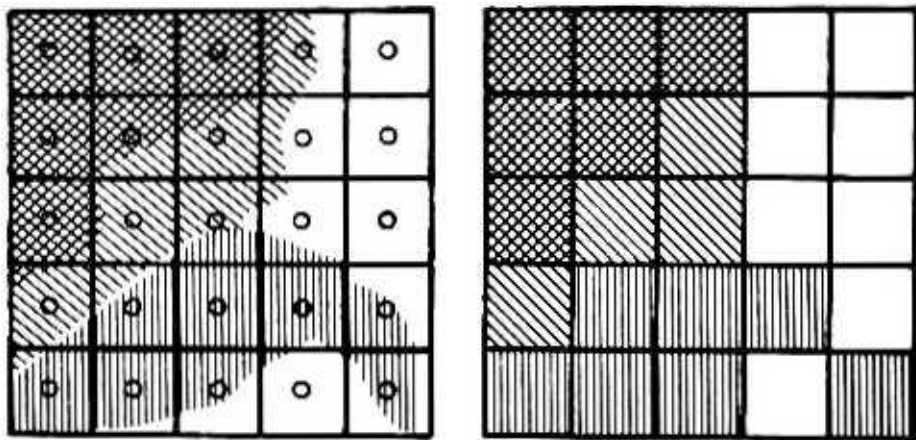


Figura 2.6 Eșantionarea bidimensională și afișarea corespunzătoare. Observați introducerea muchiilor ascuțite în Utter. Acestea corespund frecvențelor înalte introduse de eșantionare, așa cum se arată în Figura 2.1. Această figură oferă, de asemenea, o explicație detaliată a motivelor pentru aspectul slab al figurilor 2.4 și 2.5.

Grila cea mai comună folosită în procesarea imaginilor este *grila pătrată* constând din celule pătrate dispuse ca o tablă de șah. Grila *hexagonală* (Figura 2.7) este adesea discutată în literatură, dar este rar implementată. În timp ce teorema de eșantionare unidimensională se ocupă doar de dimensiunea intervalului de eșantionare, orice rezultat despre dimensiunea celulelor grilei va depinde, de asemenea, de forma acestora și de forma exactă a funcției de distribuție a punctelor. În continuare, ne ocupăm în primul rând de grile pătrate și ne referim la alte tipuri doar ocazional.

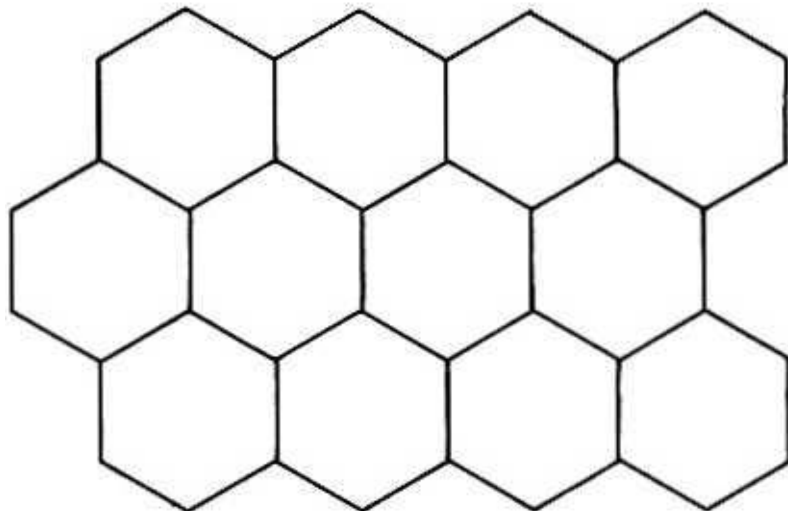


Figura 2.7 Grilă hexagonală. Jt are avantajul că toți vecinii unei celule *C* au aceeași formă de contact cu *C.* în timp ce într-o grilă pătrată trebuie să distingem vecinii care se ating pe laturi sau colțuri. De asemenea, are dezavantajul că distanța dintre centrele celulelor este diferită în direcția verticală și orizontală.

2.4 ALIASING

Dacă intervalul de eșantionare nu îndeplinește condițiile teoremei 2.1, atunci apare o distorsiune a spectrului așa cum se arată în diagrama de jos din figura 2.1. Frecvențele înalte sunt pliate pe frecvențe inferioare, producând un fenomen numit *aHasing.* Figura 2.8 arată generarea sa în cazul unidimensional: un semnal de înaltă frecvență! apare ca un semnal de joasă frecvență după eșantionarea la o rată prea mică. Un exemplu tipic în cazul datelor picturale este eșantionarea unei imagini cu text. Dacă eșantioanele sunt prea rare, rezultatul va fi un model aleatoriu de zone întunecate și luminoase, mai degrabă decât formele literelor. Un alt exemplu este digitizarea unei imagini în care aspectul nivelului de gri este produs prin variarea densității punctelor alb-negru. Dacă rata de eșantionare este comparabilă cu distanța dintre puncte, atunci valorile eșantionate vor corespunde cu alb sau negru, dar cu o distribuție diferită față de punctele originale. Aspectul rezultat poate fi foarte slab.

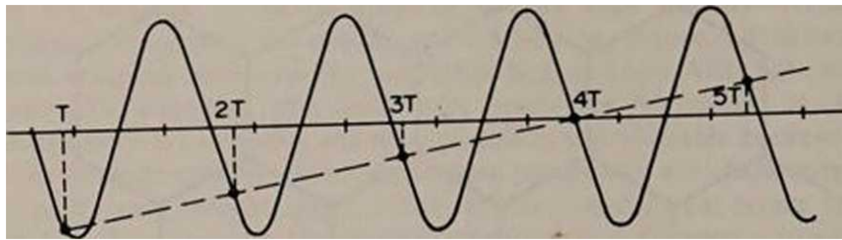


Figura 2.8 Ilustrarea aliasing-ului: linia continuă este semnalul care este eșantionat la intervale puțin mai scurte decât perioada sa. Cercurile solide arc mostrele care sunt interpretate ca reprezentând un semnal de frecvență foarte joasă (linie întreruptă).

În cazul textului, remediul este creșterea ratei de eșantionare. Un alt remediu este necesar în al doilea caz: valorile punctelor originale trebuie să fie mediate. Cu alte cuvinte, este necesar să treceți semnalele printr-un filtru trece-jos înainte de eșantionare.

Exemplul 2.2: Să presupunem că ne interesează doar frecvențele sub 1000 cps, dar semnalul conține o componentă „neinteresantă” la 1800 cps. Eșantionarea la 2000 cps va avea ca rezultat o componentă de 200 cps, care este un alias al componentei de 1800 cps. Pentru a-l elimina, semnalul original trebuie procesat de un *filtru analog trece-jos* înainte de eșantionare pentru a suprima toate frecvențele de peste 1000 cps. □

Aliasarea prezintă probleme majore în procesarea imaginii din cauza dificultății de proiectare a filtrelor analogice pentru preprocesarea semnalului. Defocalizarea sistemului optic al digitizatorului este echivalentă cu trecerea imaginii printr-un filtru simplu trece jos și acest lucru poate fi suficient în unele cazuri.

În literatura de grafică pe computer, termenul de aliasing este adesea folosit pentru a descrie distorsiunea introdusă de reconstrucțiile pe bucăți, cum ar fi cele prezentate în figurile 2.4 și 2.5. Strict vorbind, problema nu este maparea unei frecvențe înalte pe o frecvență joasă, ci introducerea unor frecvențe înalte suplimentare din cauza metodei de reconstrucție. Confuzia a apărut deoarece una dintre remediile pentru această problemă este creșterea ratei de eșantionare (Exemplul 2.1), iar același remediu este folosit și pentru a elimina aliasing-ul. Distincția dintre cele două fenomene devine clară atunci când observăm că aspectul figurilor 2.4 și 2.5 este îmbunătățit prin defocalizare, un filtru trece jos aplicat imaginii *digitale.* Un astfel de filtru nu ar putea corecta aliasing-ul decât dacă ar fi aplicat pe *imaginea analogică.*

2.5 CUANTIZAREA

Atunci când un semnal analogic este eșantionat, valorile obținute trebuie să fie reprezentate printr-un număr finit de biți, cel disponibil în computerul utilizat. Acest proces, numit *cuantizare,* poate fi considerat a fi o mapare de la numerele reale într-o serie de numere întregi. Alegerea numărului de niveluri de gri pentru a reprezenta o imagine alb-negru trebuie să țină cont de proprietățile vederii umane, dar o tratare a acestui subiect depășește sfera acestui text. Este cu siguranță adevărat că opt biți (256 de niveluri) sunt suficienți pentru majoritatea imaginilor și că deseori se pot obține afișaje bune cu doar șase biți. Pentru imagini color de bună calitate, este posibil să fie necesar să mergeți până la șase biți (64 de niveluri) pentru fiecare dintre cele trei culori de bază (în total optsprezece biți) sau chiar mai mult. Cu toate acestea, patru biți (16 nivele) pe culoare pot fi suficienți în multe aplicații. Folosirea unui octet pe pixel și alocarea a trei biți pentru două dintre culori și a doi biți pentru a treia dă rezultate marginale. Figurile 2.9 și 2.10 (Plansa 3) prezintă exemple de imagini alb-negru cuantificate la diferite niveluri. O comparație a ambelor imagini cu originalul de opt biți din Figura 2.3 (Plansa 1) arată o deteriorare a aspectului, în principal din cauza contururilor vizibile între nivelurile de gri. Cu toate acestea, există informații considerabile la nivel de gri, chiar și în imaginea pe doi biți. Calitatea unei imagini de șaisprezece nivele (patru biți) este atât de bună încât este imposibil să o distingeți de originală atunci când ambele imagini sunt tipărite pe hârtie. (Totuși, este posibil să facem diferența atunci când imaginile sunt vizualizate una lângă alta pe un monitor de televizor.)

Este posibil să îmbunătățiți aspectul imaginilor cuantificate grosier prin adăugarea de *dither.* Dacă valoarea imaginii la un moment dat (x,y) este *f (x,y)* îi adăugăm un dither *d(x,y)* înainte de cuantizare. Dither-ul *d(x,y)* poate fi generat printr-un proces aleator sau poate fi determinat din locația punctului (x,^). În niciun caz, valoarea sa nu este legată în niciun fel sistematic de *f^x.yf* Acest proces tinde să spargă contururile și, chiar dacă adăugăm zgomot imaginii, aspectul general este îmbunătățit. Figurile 2.11 și 2.12 (Plansa 4) sunt versiuni ditherate ale Figurilor 2.9 și, respectiv, 2.10 (Plansa 3) și au fost produse după cum urmează. Valoarea lui *d(x,y)* a fost luată cu probabilitate uniformă din mulțimea celor cinci numere

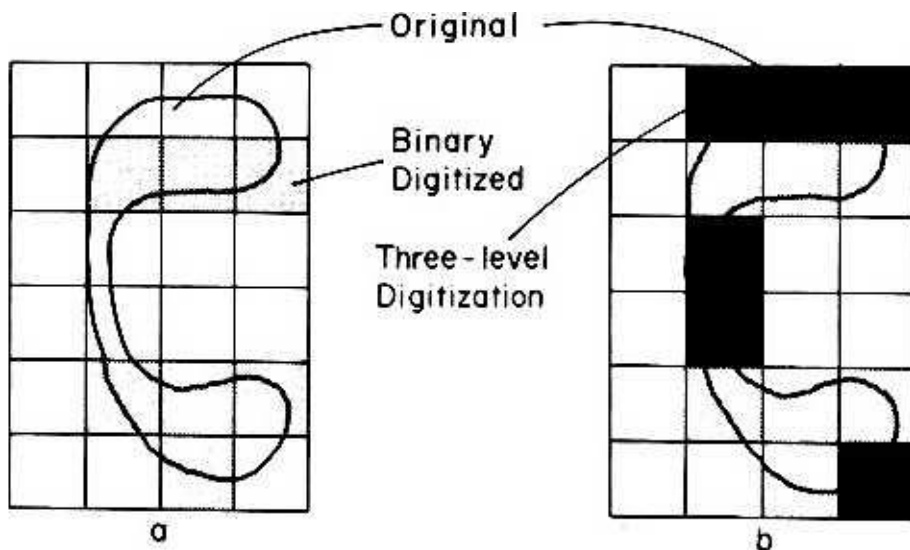
$$-2<^{w}\backslash -2^{(W)}, 0, 2^{(W)} \text{ și } 2^{(W)},$$

unde *b* este numărul de biți care trebuie afișați. Apoi, cei mai semnificativi *b* biți ai sumei au fost utilizați ca valoare a pixelului. (Dacă suma a fost negativă, a fost setată la zero înainte de mascarea biților.)

O problemă care a primit puțină atenție este compromisul dintre rata de eșantionare și nivelurile de cuantizare. Dacă avem un fix

numărul de biți disponibili pentru stocarea unei imagini date, care este cea mai bună alocare de biți pentru fiecare proces? De exemplu. 8192 de biți pot fi folosiți pentru a stoca o grilă de 64X64 (4096 pixeli), cu 4 nivele de gri (2 biți) sau o grilă de 32x32 (1024 pixeli) cu 256 de niveluri de gri (8 biți). Este evident că răspunsul depinde de tipul de imagine, dar și că pentru o anumită imagine se poate folosi eșantionarea variabilă și cuantizarea pentru diferite părți. Secțiunea 3.2 descrie câteva probleme conexe. O altă considerație este cantitatea de calcul necesară. În majoritatea problemelor costul de calcul este o funcție de numărul de mostre (pixeli), dar nu este afectat de numărul de biți pe pixel, atâta timp cât aceștia nu depășesc dimensiunea cuvântului pentru computerul utilizat. Astfel, timpul pentru procesarea unei imagini de 4096 pixeli de un bit poate fi de patru ori mai mare decât pentru o imagine de 1024 pixeli care ocupă fiecare patru biți.

Cele mai multe studii din literatura publicată sunt experimentale, iar Notele Bibliografice oferă câteva referințe înrudite. Următorul exemplu ilustrează unele dintre problemele specifice implicate.



Figare 2.13 (a) Crearea unui gol din cauza rezoluției insuficiente . (b) Creșterea numărului de niveluri de gri permite detectarea topologiei adecvate a caracterului.

Exemplul 2.3: Sarcina noastră este să digitizăm imagini cu două lcvcl, cum ar fi paginile de text tipărit. Cu toate acestea, din cauza naturii funcției de răspândire a dispozitivului de digitizare, se observă o gamă largă de valori. (Putem presupune că valoarea unui eșantion este valoarea medie a intrării pe o celulă de eșantionare.) Astfel, trebuie să ne ocupăm de o imagine de clasa 1 în timp ce intrarea a fost o imagine de clasa 2. Putem

încerca să revenim la o imagine de clasa 2 utilizând doar un bit per pixel și dându-i valoarea zero dacă intrarea este sub un anumit prag și valoarea unuia în caz contrar. În mod clar, pixelii care corespund celulelor de eșantionare din apropierea limitelor regiunii ale originalului vor fi alocați zero sau unu într-un mod mai mult sau mai puțin arbitrar, iar acest lucru poate duce la crearea sau eliminarea golurilor, așa cum se arată în Figura 2.13a. Din acest motiv, se pot alege celule de eșantionare de dimensiuni suficient de mici, astfel încât, chiar și în cele mai înguste părți ale unei regiuni, o celulă întreagă să fie cuprinsă în întregime într-o singură zonă de culoare. (Sec Capitolul 7 pentru mai multe despre acest subiect.) Alternativ, se poate alege să cuantizeze la patru niveluri (doi biți pe pixel) la o rezoluție mai mică, așa cum se arată în Figura 2.13b. □

2.6 NOTE BIBLIOGRAFICE

Majoritatea cărților despre procesarea imaginilor discută în detaliu utilizarea transformărilor și a operațiilor de filtrare liniară. Cartea lui Pratt [2.PR] este o sursă bună pentru lectură mai avansată asupra subiectelor abordate în acest capitol și în următorul capitol. Ea pune accent pe procesarea semnalului și are o discuție bună despre optică. Cartea lui Castleman (2.CA] este mai concisă, dar și cu o arie mai limitată. Hall (2.HA] și Rosenfeld și Kak [2.RK] acoperă multe dintre aceste subiecte și discută, de asemenea, subiecte legate de segmentare și recunoaștere a modelelor. Există multe lucrări care tratează aspectele practice ale cuantizării și eșantionării în imagini discrete. Imaginile eșantionate „gross” similare cu Figura 2.5 au fost discutate de Harmon și Julesz |2.HJLt [2JJN] discută unele dintre utilizările dither-ului. Aplicațiile teoriei eșantionării pentru afișarea corectă a imaginilor este un subiect care a câștigat popularitate în

♦ Dali a pictat o imagine folosind aceste concepte și aparent motivat de (opera moștenitoare. Constă din blocuri în care culoarea este aproape, dar nu exact, uniformă. Când oamenii o privesc de la distanță, văd doar blocurile și percep o imagine eșantionată „grosnoasă” a lui Lincoln. Când privitorii se apropie de imagine, atunci pot detecta detaliile imaginii din fiecare bloc și apar în fața unei ferestre ale unei femei.

literatura despre grafica pe computer. (Cititorului i se reamintește că ceea ce se numește aliasing în literatura respectivă nu este întotdeauna aliasing așa cum este înțeles în literatura de prelucrare a imaginii. Expresia informală „eliminarea jaggies” este un termen mai precis.) (2.GS) și (2.KU) sunt două lucrări recente care conțin rezultate teoretice interesante.

2.7 LITERATURA RELEVANTA

(2.CA] Castleman, KR *Digital Image Processing,* Englewood Cliffs, New Jersey: Prentice-Hall, 1979. 428 pp.

[2.GS] Gupta, S. și Sproull, RF „Filtering Edges for Gray-Scale Displays”. *SIGGRAPH 81.* Dallas, Texas (august 1981), p. 1-5.

[2.HA) Sala. EL *Procesarea și recunoașterea imaginilor pe computer.* New York: Presa Academică, 1979, 584 pp.

[2.HJJ Harmon, L. și Julcsz, B. „Masking in Visual Recognition: Effects of Two-

Dimensional Filtered Noise”, *Science,* 180 (1973), pp. 1194-1197.

12JJN1 Jarvis, JF; Judice, CN; și Ninke, WH „A Survey of Techniques for the Display of Continuous Tone Pictures on Bilevel Displays”, *CG1P.* 5 (1976), p. 13-40.

[2.KH| Knowlton, K. și Harmon, L. „Computer-Produced GreyScales”. *CGIP.* I (1972). pp. 1-20.

(2.KU) Kajiya, J. și Ullner, M. „Filtrarea textului de înaltă calitate pentru afișare pe dispozitivele de scanare raster”. *S1GGRAPH'8I.* Dallas. Texas (august 1981), p. 7-15.

[2.NL] Netravali, AN și Limb. JO „Codificarea imaginilor: o recenzie”. *IEEE Proceedings.* 68 (1980), p. 366-406.

(2.PR) Pratt. WK *Digital Image Processing,* New York: J. Wiley, 1978. 750 p.

(2.PR2) Pratt. WK (ed.) *Tehnici de transmisie a imaginilor.* New York: Academic Press, 1979, 281 pp.

[2.RKJ Rosenfeld. A. și Kak, AC *Digital Picture Processing.* New York: Academic Press. 1976, 457 p.

[28A) Sakrison. DJ „Image Coding Application of Vision Models”, *în* (2 PR2).pp 23-7).

2.8 PROBLEME

2.1. Implementați algoritmul 2.1 presupunând că există suficient spațiu pentru a stoca imaginea în memoria rapidă.

2.2. Abrogă problema 2.1 presupunând că imaginea nu se potrivește rapid

memorie. Acest lucru va necesita scrierea unei proceduri pentru a găsi transpunerea unei matrice fără a avea vreodată întreaga matrice stocată ca o matrice.

2.3. Scrieți un program pentru a implementa o reconstrucție liniară pe bucăți *(reține de ordinul întâi)* a unei imagini eșantionate. Poate doriți să utilizați următorul formalism. Dacă a, 6, c și *d* sunt valorile a patru pixeli care formează un pătrat,

ab

cd

x și > indică coordonatele relativ la colțul din stânga sus și *h* dimensiunea laturii pătratului, atunci putem defini valoarea *z a* punctelor intermediare ca

v (c—a)+x (6—a),

z - ^ -----------^— *L* --------- + *a x<hy* ..

n

y(db)+x(dc), ., .

z - *1* ------ - + *c+b~d x>hy*

(Aceste ecuații pot fi derivate din geometria elementară sau din ecuația (16.2).)

2.4. Utilizați programul de mai sus pentru a studia compromisurile dintre eșantionare și cuantizare. (Puteți dori să scrieți aceste programe ca parte a unui editor de imagini. Vezi următoarea problemă.)

2.5. Îmbogățiți editorul de imagini al Problemei 1.5 prin adăugarea de caracteristici care vă permit să studiați variațiile ratei de eșantionare sau numărul de niveluri de cuantizare, precum și compromisurile dintre cele două.

Sugestii. Puteți implementa comanda EXPAND și studiați variațiile ratei de eșantionare luând un subset de pixeli ai imaginii originale pentru a crea una nouă și apoi afișați-o la aceeași dimensiune prin extindere. Puteți alege să adăugați două comenzi noi în meniu: SAMPLE și LEVEL. De fiecare dată când primul este executat, acesta ia fiecare alt pixel în direcțiile verticală și orizontală (unul din patru) și apoi folosește extinderea pentru a afișa rezultatul. LEVEL golește biți începând de la sfârșit. Sau puteți alege comenzi care acceptă argumente numerice care specifică cu precizie rata de eșantionare și numărul de niveluri de cuantizare.

2.6. Analizați schema de dithering discutată în Secțiunea 2.5.

ANEXA 2.A: TRANSFORMĂ RAPIDĂ DE FOURIER

Oferim aici o scurtă trecere în revistă a transformării rapide Fourier. Cititorul poate consulta un text despre procesarea semnalului sau algoritmi pentru o analiză mai detaliată. Punctul nostru de pornire este ecuația (2.3) care înlocuiește exponențialul cu z așa cum este definit în secțiunea 2.2.

FM-^/(k)!* 0<u<Nl* (2.AI)
*-o

Introducem un nou indice de însumare, *m,* care variază de la 0 la (/V/2)-l și care este legat de *k* prin următoarele ecuații.

k = 2m	*bifurcație* pară	(2.A.2a)
*-2m+l	pentru* impar	(2.A.2b)

Apoi ecuația (2.A. 1) poate fi rescrisă ca

(/V/2J-l
F(«)= 2 *[/(.imyz^ +f(2m+l)z^ +, >'] .* (2.A.3) •*-0

Să definim acum *M = N/2* și

g =cxp(-;-~-)

și observați că z ^{2} - *g.* Atunci ecuația (2.A.3) poate fi scrisă ca

MI MI
FGO“ *2fAn)g™ +z2f* $_o$ *(r* $_n$ *)g™* (2.A.4) M-0 ".-0

unde *f* $_r$ și *f* $_0$ sunt două funcții noi obținute luând în considerare separat punctele pentru valorile pare și impare ale argumentului. Cele două sume din ecuația (2.A.4) seamănă foarte mult cu transformatele Fourier ale funcțiilor cu *M* puncte, cu excepția faptului că *u* variază de la 0 la ^“1. Cu toate acestea, observăm că

F(u+N)-F(u) (2.A.5)

întrucât exp(-//V-^-) = 1. Astfel, valorile *M* ale transformării Fourier a lui A (sau A) pot fi folosite pentru a găsi *N* valori trivial. Prin urmare, wc are următoarea ecuație *recursivă* pentru transformata Fourier.

^(«î-FJuJ + expC-y^FUu) (2.A.6)

Ecuația (2.A.6) este baza transformării rapide Fourier. În primul rând, oferă o metodă ușor de programat pentru calcul.

Acesta este dat ca algoritmul 2.AI

Algoritmul 2.A.1 Transformată Fourier rapidă
Prmttun FFT(NJ)

1. Dacă /V este egal cu 2. atunci faceți:
ÎNCEPE.

2. Înlocuiți/(O) cu/(O) + /(l)și/(l) cu/(O) —/(I).
3. Reveni.
Sfârşit.
4. Altfel face:
ÎNCEPE.
5. Definiți *g* ca fiind format din toate punctele lui / care au un indice par și *h* ca fiind format din punctele rămase.
6. Procedura de apel *FFT(N/2,g).*
7. Procedura de apel *FFT(N/2,h).*
8. Înlocuiți *f(i)* cu *g(i) + e*p(-j2ir/N)h{i)* pentru i-0 la /Vl.
Sfârşit.
9. Sfârșitul procedurii

Procedura se numește singură dacă *N* depășește 2, în timp ce pentru TV — 2 folosește formula

f(0)=/(0)+/(l) (2.A.7a)

F(IJ =/(0)-/(l) (2.A.7b)

În mod clar, înjumătățirea succesivă a lui *N* poate decurge fără probleme numai dacă N este o putere a lui 2 și doar pentru astfel de valori există un algoritm simplu pentru transformarea Fourier rapidă. În al doilea rând, ecuația (2.A.6) poate fi utilizată pentru a calcula costul calculului. Fie C(/V) costul pentru *N* puncte. După evaluarea celor două transformări din partea dreaptă, ecuația (2.A.6) necesită efort proporțional cu *N* din cauza înmulțirii termenilor cu exponențial și a adunării ulterioare . Dacă c este o constantă care reflectă costul unor astfel de operațiuni, atunci avem următoarea ecuație pentru *C(N).*

C(Nf-2C{^ + cN (2.A.8)

În mod similar, găsim că

C(y) - *2C^) + c ^* (2.A.9)

şi aşa mai departe. Datorită ecuațiilor (2.A.7) știm că C(2) este egal cu costul a două adunări, pe care le notăm cu *c'.* Dacă scriem ecuații

similar cu ecuația (2.A.9) pentru C(-y-), unde £ = 4, • • • , *N/4,* înmulțiți fiecare dintre ele cu *L,* adăugați-le la ecuația (2.A.8). și adunăm ecuația (2.A.9) înmulțită cu două, aflăm că

CW^c'+cNn (2.A.10)

unde *n* este numărul acestor ecuații. În special *n* este egal cu numărul de termeni din seria 1. 2, 4, • ■ • . *N/^* care este egal cu logaritmul în baza 2 al lui *N.* Iog 2 /V, minus I. Atunci ecuația (2.A.10) devine

C(/V) -cNlog2A < +O(M (2.AH)

unde ultima expresie denotă termeni liniar proporționali cu *.N.*

Capitolul 3

PRELUCRARE DE IMAGINI ÎN SCANĂ DE GRI

3.1 INTRODUCERE

Există două tipuri majore de procesare a imaginilor la scară de gri: transformările în cadrul clasei, cum ar fi *filtrarea* și *îmbunătățirea imaginii,* și transformările de imagine de clasa I în clasa 2, cum ar fi *segmentarea.* Majoritatea metodelor de efectuare a unei astfel de prelucrari folosesc, direct sau indirect, statistici calculate pe imagini. Vom discuta două dintre ele, *histograma* de distribuție a nivelurilor de gri (în Secțiunea 3.2) și *matricea de co-ocurență a perechilor de niveluri de* gri la perechile de pixeli (în Secțiunea 3.3). Aplicațiile lor în filtrare sunt discutate în Secțiunile 3.4 și 3.5, în timp ce utilizarea lor pentru segmentare este prezentată în capitolul următor.

Dacă tratăm o imagine ca un proces aleatoriu, atunci trebuie să definim funcția de densitate de probabilitate *de ordinul întâi p^PZ)* că un pixel *P* (specificat de locația sa) are nivel de luminozitate sau culoare *Z.* Pentru imaginile monocrome vom presupune întotdeauna că *Z* variază între 0 (foarte întunecat) și o valoare £ >0 (foarte luminos). Dependența acestei probabilități de locația unui pixel permite utilizarea modelului pentru a genera imagini semnificative.

Exemplul 3.1: Luați în considerare o imagine care conține un caracter întunecat pe un fundal deschis. Fie C_0 mulțimea de pixeli care aparțin unei forme ideale a caracterului și au valoarea *D.* Pixelii de fundal au valoare

3.2 HISTOGRAMA SI EGALIZAREA HISTOGRAMA

Algonthmul 3.1 arată cum se calculează o estimare a *p^P ,Z}* dintr-o singură imagine, în ipoteza că aceasta este independentă de *P.* Rezultatul este cunoscut sub numele de *histograma H(Z)* a imaginii. Dacă *f (P)* indică nivelul de luminozitate la pixelul *P,* atunci Algonthmul 3.1 poate fi utilizat pentru a evalua *H{Z).* Histograma din figura 2.3 este prezentată în figura 3.2.

Algoritmul 3.1 Evaluarea histogramei

Notație: f (P) este valoarea pixelului *P* cu interval (OL|. *H* este matricea histogramei.

1. Inițializați matricea *H(Z)* (0<Z <£) la zero.
2. Pentru toți pixelii *P* ai imaginii faceți:
   ÎNCEPE.
3. Creșteți *H(J(P))* cu I.
   Sfârşit.
4. Sfârșitul algoritmului.

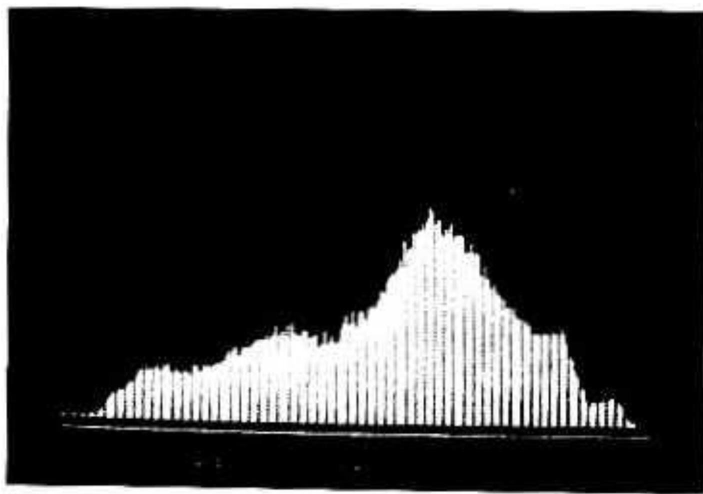


Figura 3.2 Histograma nivel de gri pentru Figura 2.3. Rețineți că unele niveluri lipsesc. Acestea lipsesc și din imaginea originală din cauza unei probleme la convertorul A/D în timpul digitizării. Unul dintre biți nu a fost niciodată setat și, prin urmare, unele niveluri de gri nu au fost realizate.

Histograma poate fi utilizată pentru îmbunătățirea sau codificarea imaginii în felul următor. Este posibil ca *H(Z)* să fie zero pentru multe valori de Z. iar aceasta înseamnă că nivelurile disponibile de cuantizare nu sunt utilizate eficient. Atunci ar fi de dorit să le reatribuiți astfel încât intervalul dinamic al imaginii să crească. Figura 3.3 (vezi Planșa 5) prezintă o fotografie subexpusă și histograma corespunzătoare. Există puțini pixeli cu valori între 180 și 255. S-ar putea încerca să

scalați valorile pentru a ocupa întreaga gamă, dar acest lucru nu este întotdeauna fezabil și cu siguranță nu pentru exemplul actual. O modalitate mai bună de a utiliza intervalul de afișare disponibil este realocarea valorilor astfel încât histograma rezultată să fie cât mai plată posibil.

Fie ^ zona imaginii și *N* numărul de niveluri de luminozitate disponibile. Pentru o histogramă perfect plată, trebuie să aveți *A/N* pixeli la fiecare nivel. Dacă valoarea luminozității unui nivel Z este de 4 ori mai mare decât media . atunci acel nivel trebuie mapat în *k* niveluri diferite de la. cuvânt. Z, până la z4 . Prin urmare, trebuie să introducem o regulă pentru realizarea acestei mapări unu-la-mulți. Nu există o politică generală care să funcționeze bine în toate aplicațiile, dar următoarele sunt trei posibilități.

Regula I Mapați întotdeauna Z pe nivelul mijlociu (Z|+ZJ/2. (Acest document nu are ca rezultat o histogramă plată, ci una în care nivelurile de luminozitate sunt distanțate.)

Regula 2 Alocați la întâmplare unul dintre nivelurile din interval (Z^Z*). (Acest lucru poate duce la pierderea contrastului dacă histograma originală a avut două vârfuri distincte care erau foarte îndepărtate.)

Regula 3 Examinați vecinătatea pixelului și atribuiți-i un nivel de la [Z|.Zj care este cel mai apropiat de media vecinătății (Acest lucru poate provoca murdărirea marginilor și necesită mult mai multe calcule decât celelalte două reguli.)

Prima regulă este pur euristică. Nu produce histograme cu adevărat egalizate, dar realizează utilizarea deplină a intervalului dinamic cu mai puțin efort decât celelalte reguli. (Într-o anumită implementare, a doua regulă a necesitat aproximativ de patru ori mai mult timp decât prima, din cauza necesității de randomizare.) Motivația pentru a doua regulă este dorința de a evita orice eroare sistematică: atunci când trebuie făcută o alegere arbitrară, faceți-o într-un mod aleatoriu. A treia regulă încearcă să impună o anumită coerență între nivelurile de pixeli, h presupune că, cu cât doi pixeli sunt mai aproape de plan, cu atât este mai probabil ca aceștia să aibă valori similare. (Consultați Secțiunea 3.4 pentru mai multe despre acest punct.)

Algoritmul 3.2 Egalizarea histogramei

Notație: H este matricea histogramei. *H^* este o integrală a histogramei. *Z* indică nivelurile vechi, iar *R* pe cele noi. Fiecare Z este mapat pe un interval *[stânga (Z).dreapta (Z)].* .

0. Citiți o imagine, evaluați-i histograma și stocați-o în tabloul *H.* Fie *H^* valoarea sa medie.

1. Setați *R* - 0 și H_M = 0.
2. Pentru Z - 0 la L faceți: Începeți.
4. Setați *stânga(Z) - R* și adăugați H(Z) la *H,*.*
5. În timp ce H_M este mai mare decât *H^* face:
   ÎNCEPE.
6. Scădeți *H^* din H_M și creșteți *R.*
   Sfârşit.

7. Setați *dreapta (Z) = R* și definiți valoarea lui *new(Z)* conform regulii utilizate. Pentru regula 1, setați *nou(Z)* la media *stânga(Z)* și *dreapta (Zf* Pentru regula 2 setați *nou(Z)* la diferența *dreapta (Z)-lefi(Z).* (Pentru Rulc * ww(Z) este 1^' nedefinit.)
Sfârşit.
8. Pentru toți pixelii *P* ai imaginii, faceți: Începeți.
9. Dacă *lefi(f (P))* este egal *cu dreapta (f(P)),* atunci setați noua valoare a pixelului *P la stânga (J (P)).*
10. Altfel:
Dacă se folosește regula 1, setați valoarea la *new(f (P)).*
Dacă se folosește regula 2, alegeți un punct la întâmplare din intervalul [0,WH(/ (P))), adăugați valoarea lui la *lefi(f(P))* și utilizați rezultatul pentru noua valoare a lui *P*
Dacă se folosește regula 3, se calculează media vecinătății lui *P.* Dacă depășește *dreapta (Z),* se folosește *dreptul (Z)* ca nouă valoare. Dacă este sub *stânga(Z),* utilizați *stânga(Z)* ca nouă valoare. Altfel , utilizați media ca nouă valoare.
Sfârşit.
11. Sfârșitul algoritmului.

Algoritmul 3.2 implementează egalizarea histogramei pentru oricare dintre aceste reguli. Pasul 0 pregătește histograma conform algoritmului 3.1. Pașii de la 1 la 7 fac egalizarea prin maparea vechilor niveluri de luminozitate pe noile niveluri. Pașii de la 8 la 10 calculează noua imagine. Figura 3.4 (Plansa 6) prezintă rezultatele aplicării sale la imaginea din Figura 3.3.

Acest tip de procesare nu poate fi aplicat imaginilor în mod indiscriminat, - deoarece egalizarea poate provoca adesea o deteriorare a

aspect. Figura 3.5 (Plansa 7) prezintă o imagine și histograma acesteia, în timp ce Figura 3.6 (Planca 8) arată rezultatele egalizării. Deoarece histograma bimodală s-a răspândit, imaginea egalizată are un aspect „murdar”.

Egalizarea histogramei poate fi, de asemenea, utilizată pentru a produce o cuantizare mai grosieră a unei imagini. De exemplu, este posibil ca o imagine să fi fost digitalizată cu multe niveluri de gri (de exemplu, 256), dar trebuie să fie afișată pe un dispozitiv cu doar opt niveluri disponibile. Algoritmul 3.2 poate fi utilizat cu *L* egal cu noua luminozitate maximă (7 în acest caz). Aceeași valoare ar trebui folosită și pentru a găsi media histogramei *H^.* Rețineți că în acest caz este puțin probabil ca un singur nivel să fie mapat în mai multe, astfel încât pașii 7 și 10 ai algoritmului pot fi simplificați. Figura 3.7 (Planca 9) prezintă rezultatele unei astfel de cuantizări pentru imaginea din Figura 2.3. O comparație între Figura 3.7 (Plansa 9) și Figurile 2.9 și 2.10 (Plansa 3) arată clar că luarea în considerare doar a celor mai semnificativi biți nu este soluția optimă. În special, pot exista aplicații în care costul unui preprocesor pentru egalizarea histogramei poate fi mai mic decât costul memoriei suplimentare de reîmprospătare necesare pentru afișajele cu mai mulți biți. Dacă factorul limitativ nu este costul memoriei, ci caracteristicile optice ale suportului de afișare, atunci o funcție de tipul *culoare (uv)*

descrisă în Secțiunea 1.7 (Tabelul 1.2) poate fi utilizată pentru a implementa egalizarea.

O problemă asociată apare în utilizarea *pseudocolorului.* În unele aplicații, detectarea informațiilor importante necesită discriminarea între nivelurile de gri care se apropie unele de altele. Este posibil să utilizați egalizarea histogramei pentru a extinde gama de intensități utilizate în regiunea de interes. Alternativ, putem mapa astfel de niveluri nu pe o scară de gri, ci pe *culoare.* Deoarece numărul de culori distincte disponibile poate fi mic, trebuie să folosim egalizarea histogramei și în acest scop.

Exemplul 3.3: În multe probleme de diagnostic medical sau de control al calității industriale, cineva este interesat de detectarea petelor întunecate într-o imagine radiografică. Forma unor astfel de pete este importantă pentru diagnostic, astfel încât acestea trebuie preluate cu atenție din fundal. Dacă te uiți la imaginea originală, astfel de pete „patologice” pot fi confundate cu „umbre”. O formă de îmbunătățire a imaginii este creșterea rezoluției luminozității pentru părțile mai întunecate ale imaginii. Dacă intervalul datelor originale este (0,255), atunci o astfel de îmbunătățire poate fi realizată prin maparea intervalului (0,31) pe (0,254) (folosind algoritmul 3.2 cu Regula 1) și intervalul (32255) la valoarea 255. Valorile luminozității ar trebui să difere acum una de cealaltă cu aproximativ zece procente sau mai mult. Un aranjament alternativ este de a lua în considerare histograma în intervalul (0,31]

numai, mapați-o pe opt valori și apoi atribuiți fiecăreia dintre ele la una dintre cele opt culori printr-o instrucțiune *de culoare (uv)* (vezi Tabelul 1.2). □

3.3 MATRICE DE CO-OCURREnță

O *matrice de co-ocurență* este o estimare a $p_2(P, Q. Z, Y\}$ asupra unei singure imagini în ipoteza că p_2 depinde doar de poziția relativă a lui *P* și *Q.* O astfel de matrice o notăm cu $C,(Z,Y)$ unde *r* reprezintă relația dintre *P* și *Q.* Numărul de astfel de matrici poate fi destul de mare și poate fi destul de mare și numai în ordinea unor simplificări ulterioare. al unui pixel: unul deasupra acestuia sau unul la dreapta acestuia. În al doilea rând, se poate ignora complet orientarea și se face media matricelor obținute pentru diverse orientări.

Algoritmul 3.3 Evaluarea Matricelor de Co-ocurență.

1. **Pentru** toate relațiile *r* dintre perechile de pixeli faceți:

   ÎNCEPE.

2. Inițializați matricea $C,(Z,Y)$ $(Q \hat{} Z \hat{} L, Q \hat{} Y < L)$ la zero.
3. **Pentru** toți pixelii *P* ai imaginii **faceți:**

   ÎNCEPE.

4- Fie *Q* să desemneze vecinul lui *P* conform relației

   r. Creșteți $C,(f(P)J(Q))$ cu 1.

Sfârşit.

Sfârşit.

s. Sfârșitul algoritmului.

Este posibil ca dacă P este un pixel în apropierea cadrului imaginii, pixelul *Q* definit de relația *r* să nu existe. În acest caz, putem înlocui *f^Q}* cu un zero sau cu o altă valoare specială. În continuare, wc va presupune că pixelii *P* și *Q* sunt întotdeauna adiacenți. De asemenea, în exemplele acestei secțiuni wc va lua în considerare doar două relații: când *Q* este la dreapta *lui P* (relația 1). iar când *Q* este peste *P* (relația 2). Multe proprietăți ale matricei *C* sunt evidente din modul în care este construită. În primul rând, elementele sale diagonale sunt aproximativ egale cu ariile regiunilor, cu elementul $C_r(kk)$ egal cu aria regiunilor a căror

pixelii au valoarea *k.* În al doilea rând, elementele off diagonale au valori aproximativ egale cu lungimea granițelor dintre regiuni, cu elementul *Cr(kJ)* egal cu lungimea conturului dintre regiunile ai căror pixeli au valori *k* și *j.* Pentru imaginile cu contrast scăzut, elementele departe de diagonală ar trebui să fie zero sau foarte mici, în timp ce opusul va fi valabil pentru imaginile cu contrast ridicat. O dificultate majoră în utilizarea matricelor de co-ocurență este dimensiunea lor mare *(LXL).* Este posibil să se depășească această problemă prin introducerea unei cuantizări mai grosiere în timpul estimării matricei. Tehnica bazată pe egalizarea histogramei . descrise în secțiunea anterioară, pot fi utilizate în acest scop.

Exemplul 3.4: Să presupunem că în imaginea din Exemplul 3.2 fiecare bandă are cinci pixeli lățime, imaginea este pe două niveluri și are dimensiunea *NXN.* Atunci cele două matrice de co-ocurență vor fi (aproximativ)

^IZZI <'■'*)

C >' [Vw2| < 3 lb >

Rețineți că aceste ecuații ar rămâne aceleași dacă imaginea ar fi schimbată atâta timp cât numărul de dungi și lățimea lor medie ar rămâne neschimbate. □

Trei exemple suplimentare de matrice de co-ocurență, calculate pe părți din Figura 2.3. sunt prezentate în Figura 3.8. Pentru a reduce dimensiunea matricei a fost utilizată o cuantizare cu opt niveluri. Nivelurile au fost găsite prin egalizarea histogramei pe întreaga imagine, astfel încât rezultatul a fost acelaşi ca în Figura 3.7b (Plansa 9). Matricea (a) a fost evaluată pe un pătrat de 128X128 pe perdea, matricea (b) pe un pătrat de 90X90 pe păr și matricea (c) pe un pătrat de 80X80 pe cămașă. Fiecare matrice este media pentru patru relații: deasupra, dedesubt, stânga și dreapta. În mod clar, aceste matrici oferă informații despre *textura,* precum și despre luminozitatea medie a imaginii. Deși textura este un termen pe care oamenii îl înțeleg ușor, definiția sa formală este mult mai grea. Se poate presupune că, dacă două zone ale unei imagini au același set de matrice de co-ocurență, atunci ele vor părea, de asemenea, să aibă aceeași textură (vezi Notele bibliografice).

	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	353	444	17	0	0	0
4	0	0	444	5791	1902	595	17	0
5	0	0	17	1902	2452	2077	138	0
6	0	0	0	595	2077	5970	1812	5
7	0	0	0	17	138	1812	3676	51
8	0	0	0	0	0	5	51	17

(o)

	1	2	3	4 5 6 7 8				
1	483	422	9	8	0	0	0	0
2	422	7350	1450	71	0	0	0	0
3	9	1450	3068	413	1	0	0	0
4	8	7)	413	267	4	0	0	0
	0	0	1	4 0 0 0 0				
5 6	0	0	0	00000				
7	0	0	0	00000				
8	0	sau	0	00000				

(b)

	1	2	3	4	5	6	7	8
1	650	127	31	32	21	29	28	9)
2 3	127	535	124	55	18	21	41	48
	31	124	435	142	33	39	51	110
4	32	55	142	667	113	88	85	172
5 6	21	18	33	113	253	179	55	108
7 8	2'	21	39	88	179	317	111	201
	2«	41	51	85	55	III	109	338
	91	48	IIO	172	108	201	338	4589

(c)

Figura 3.8 Trei matrici de co-ocurență

3.4 FILTRAREA LINEARĂ A IMAGINILOR

O cantitate semnificativă de procesare a imaginii poate fi efectuată fără a anula analiza statistică din secțiunea anterioară pentru fiecare imagine nouă. Doar unele cunoștințe a *priori limitate sunt suficiente.* De exemplu, să presupunem că forma matricei de co-ocurență a unei imagini) este cunoscută și că dorim să îmbunătățim o copie zgomotoasă a originalului . Dacă matricea are cele mai mari demente de-a lungul sau lângă diagonala principală , atunci știm că majoritatea pixelilor ar trebui să aibă culoarea vecinilor lor. Dacă am dori să egalăm histograma unei astfel de imagini, atunci am fi justificați să folosim Regula 3, așa cum sa discutat în Secțiunea 3.2. Dacă dorim să eliminăm zgomotul, atunci înlocuirea fiecărui pixel al imaginii zgomotoase cu o sumă ponderată a vecinilor săi va reduce variabilitatea între pixelii adiacenți și vom obține o imagine mai apropiată de originală (vezi Exemplul 3.5 de mai jos). În acest fel. suntem conduși la o ecuație care descrie relația dintre imaginea originală *f(xy)* și imaginea *filtrată g(xy):*

*Ud')- £ $ *h(xj.iJ)f(x+i* $_t$ *y+J')* (3.2) *i--M j--M*

Procesul care efectuează această operație se spune că este o *fibră liniară,* și în special un filtru *de medie mobilă* , deoarece la fiecare pixel îi înlocuim valoarea cu un fel de medie a valorii vecinilor săi . Dacă funcția de ponderare A este aceeași în imagine și nu depinde de (x^), atunci ecuația (3.2) poate fi scrisă ca

*g(x,y)^* 2 % *h(ij)f(x+iy+j)* (3.3)
Eu--M)-~M

Acesta se numește filtru *invariant de spațiu* . Astfel de filtre sunt comune în procesarea semnalelor de timp, dar relevanța lor pentru procesarea imaginilor este discutabilă. Ecuația (3.3) ia o formă mai simplă atunci când este scrisă în termeni de transformate Fourier. Se poate arăta (vezi Notele Bibliografice) că

G(uv) -W(uv) F(u,v) (3,4)

Efectul filtrului este de a atenua anumite frecvențe și de a amplifica altele, în funcție de mărimea lui//(«,v).

Exemplul 3.5: Dacă se dorește eliminarea zgomotului de înaltă frecvență dintr-o imagine, atunci o astfel de netezire poate fi realizată folosind, printre altele, următoarea formă de *h(ij):*

A (0,0)-j (3-5a)

H1,O) = A (0,1) = *(0,-1) = M-1,0) = y (35b >

MU) = *0 -1) = *(-iD = MJ -D =-**y** < 35c >

Efectul filtrului poate fi văzut comparând diferența dintre valorile pixelilor vecini înainte și după aplicarea acestuia. În special, un calcul simplu arată că guj-)-g(x+u> = -

^/(^jo-yu+u)! +

v(/(*->^)-/(x+>o')l + o

-rr(ruj'-i)+/(*^+i)-/u+i0'-i)-/u+ij-+i)+

/(xl^-|)+/^-l.>'+l)-/(x+2^-|)-/ (x+2j-+l)l (3.6)

Dacă *Df* reprezintă diferența absolută maximă dintre pixelii adiacenți din *f(x,y)* și *D g* diferența corespunzătoare în ^(x^), atunci ecuația de mai sus implică faptul că

°« -1°' + f^ + 17 4 2 ^ " D > (37)

adică acea diferență nu poate crește. Egalitatea apare numai atunci când diferența maximă dintre *n* pixeli este *de n* ori diferența dintre o pereche, adică atunci când *f (x,y)* este o funcție liniară a argumentelor sale. Dacă acest lucru nu este adevărat, atunci diferența va scădea și regiunile vor deveni mai uniforme. De obicei, deoarece un astfel de filtru simplu nu este suficient pentru eliminarea zgomotului, trebuie să folosiți un filtru de ordin superior. O posibilă implementare a filtrelor de ordin superior este utilizarea unui filtru simplu și aplicarea acestuia în mod repetat imaginii. Figura 3.10 (Placa II) prezintă rezultatele unei astfel de prelucrări: (a) este imaginea originală care a fost obținută din Figura 3.9 (Placa 10) prin adăugarea de *zgomot alb gaussian* și (b) este rezultatul obținut după opt aplicații ale filtrului ecuației (3.5). Se poate observa, imediat, că acest proces îndepărtează nu numai zgomotul de înaltă frecvență, ci și pete marginile. Într-adevăr, dacă aplicăm ecuația (3.6) unei muchii perfecte

/(xy) = I pentru x <%

/(x^) = 0 pentru *x>X*

găsim

«Uj')-«(A' + l^)-| + 4- + ^“voo io z

Cu alte cuvinte, diferența dintre valorile pixelilor adiacenți a fost redusă la jumătate și acest lucru a dus în mod clar la pierderea contrastului.

Putem studia și efectele filtrului în ceea ce privește transformarea Fourier a lui *h(ij).* Fie £(x) o abreviere care indică termenul

£(x) = exp(j—x) .

Atunci prin aplicarea ecuației (3.4) rezultă „(“• v > = 7 + j^(-^)+£(v)+£(-u)+£(u)j +

-^t(-(u+v))+£(-(uv))+£(uv)+£(u+v)

(38)

Această ecuație poate fi simplificată folosind mai întâi COSTUL DE IDENTITATE = |[exp(jr)

+ exp(-»|

și apoi transformări trigonometrice simple astfel încât să ia forma *H(u,v)* = y(l + cos^-

v+cos-^-u + cos^-vcos-^-u) (3.9)

Observăm că *H* este zero ori de câte ori w sau v este egal cu *N/2* și, de asemenea, că

H(0.0)=l H(0.-7-)=W(4.0)=4 W <-T'V>“T-
4 4 2 4 4 4

Astfel, frecvențele înalte sunt atenuate. □

Această eliminare simultană a zgomotului și murdărirea marginilor sugerează că filtrele liniare, invariante în spațiu ar trebui utilizate cu prudență în procesarea datelor picturale. Din păcate, utilizarea extinsă și cu succes a filtrelor similare pentru funcțiile de timp a încurajat aplicarea acestora în procesarea imaginilor, fără a ține cont de limitările lor.

O problemă similară există pentru *filtrele de trecere înaltă.* Acestea produc imagini cu margini mai clare, dar amplifică și zgomotul de înaltă frecvență.

Exemplul 3.6: Un filtru trece-înalt simplu are forma

6(0,0)-6 (3.10a)

6(1,0) - 6(-l,0) = 6(0,1) - 6(1,0) - -a (3.10b)

Transformarea sa Fourier este egală cu

H(uv) - *b* - 2a (cos~~v +cos-^u) (311)

Valoarea maximă a lui *H* este 6+4o și se realizează când u - v = *N/2* $_y$ în timp ce minimul său este 77(0,0) — 6—4a. □

3.5 FILTRARE NELINEARĂ A IMAGINILOR

Filtrele care nu untează marginile, ci doar elimină zgomotul din interiorul regiunilor, sunt mult mai complexe decât oricare dintre formele date de ecuația (3.3). Astfel de filtre trebuie să încerce să detecteze marginile înainte de a aplica o funcție de netezire. Deoarece detectarea marginilor este dificilă pe o imagine zgomotoasă, problema nu este deloc una ușoară. Prezentăm aici câteva dintre cele mai comune abordări.

3.5.1 Filtre direcționale

O tehnică folosește un filtru liniar care este simetric în raport cu o axă, mai degrabă decât cu un punct. Apoi, la fiecare pixel, se face un efort pentru a estima direcția unei margini, dacă există, iar filtrul evită media punctelor peste margine. Această idee poate fi implementată într-un mod simplu prin definirea următoarelor două funcții ale unghiului 0 pe care îl formează o muchie cu o axă de coordonate.

0	O	45'	90'	135'
C(0)	1	1	0	-1
^(0)	0	1	1	1

Apoi funcția de filtru *hfjjrf)* este definită ca

A(O,O,0)-O.5, A(c(0),^(0),0)-A(-c(0)-j(0),0)=O.25 și zero pentru toate celelalte

argumente *ij.* La fiecare pixel valorile lui ^(0) = 1/ u^j-zu+cf^jj'+sc^)] ^{2} +

l/(XvF)-/(xc(0)),^-^(0))l ^{2}

sunt calculate pentru 0 = 0, 45*, 90’ și 135*. Apoi se aplică filtrul pentru acea valoare de 0 pentru care K(0) este minim. Efectul său este prezentat în Figura 3.11 (Plansa 12).

3.5.2 Filtre din două părți

Filtrele din două părți sunt o extensie a filtrelor liniare direcționale. Mai întâi, se aplică un filtru trece jos pe imagine, apoi se calculează gradientul lui *f(^{x} ^y)* . Acest lucru ar trebui să dea o idee despre locația marginilor. Apoi se face o a doua trecere peste imaginea filtrată folosind un filtru ai cărui coeficienți depind de locație, astfel încât marginile să nu fie murdare. Dacă imaginea originală a fost pierdută, atunci un filtru trece-înaltă poate fi utilizat pe imaginea filtrată cu coeficienți mari în zonele cu gradient mare și coeficienți mici în altă parte. Acest lucru ar trebui să restabilească valorile imaginii originale în locațiile în care gradientul are un nivel ridicat

valoare. Dacă imaginea originală este încă disponibilă, atunci un filtru trece jos ar putea fi adecvat. Există diverse perfecționări ale acestei abordări, iar cititorii se pot gândi la altele singuri (vezi Notele bibliografice).t

3.5.3 Filtre de aproximare funcțională

Acestea se bazează pe înlocuirea lui *f(x,y)* cu o estimare locală a unora dintre statisticile sale. Este necesar să împărțiți zona unei imagini în regiuni de dimensiune fixă, astfel încât aceste statistici să poată fi găsite. În cea mai simplă formă a metodei se evaluează media și varianța lui *f (x,y)* pe fiecare astfel de regiune. Dacă varianța este sub un anumit prag stabilit , atunci valorile lui *f(x,y)* sunt înlocuite cu media. În caz contrar, regiunea este împărțită în părți mai mici, iar procesul se repetă. În acest fel marginile vor fi izolate. În loc să estimăm media, se poate aproxima *f (x,y)* cu o funcție netedă și apoi se subdivizează regiunea dacă eroarea aproximării este prea mare. (Vezi capitolul 6 pentru structurile de date și capitolul 12 pentru tehnicile de aproximare.) Această abordare este, de asemenea, strâns legată de segmentare, despre care vom discuta în capitolul următor.

3.6 NOTE BIBLIOGRAFICE

Textele [2.CA], [2.HA], (2.PR] și [2.RK] citate în capitolul anterior discută problemele de filtrare și detecție a marginilor.

Dovada ecuației (3.4) poate fi găsită în majoritatea textelor elementare despre procesarea semnalului, de exemplu (3.ST] sau (3.OS). Pentru mai multe informații despre filtrele neliniare, trebuie să consultați literatura periodică . A se vedea [3.NM] pentru filtrele direcționale și [3.AS] pentru un filtru din două părți. Histogramele sunt discutate în procesarea imaginii, [PA3] și recunoașterea textului virtual. (2.PR], [2.RK], etc.). Conexiunea dintre matricele de co-ocurență și textură a fost explorată pentru prima dată de Haralick. Vezi lucrarea sa recentă [3.HA] pentru o discuție amănunțită a subiectului texturii [3.J1], [3J2] a studiat discriminarea modelelor aleatorii de ordinul lor și textura.

Munca teoretică în procesarea imaginilor se confruntă cu un obstacol serios. Lipsa unei măsuri matematice care să reflecte diferența subiectivă în aspectul imaginilor. De exemplu, să presupunem că noi

t Termenul „în două treceri” ar putea fi aplicat unor astfel de filtre, dar preferăm „în două părți” deoarece se folosește un filtru diferit în fiecare trecere. Astfel, putem distinge aceste filtre neliniare de cele liniare în care trecerile suplimentare sunt mijloace simple pentru implementarea filtrelor de ordin înalt.

au o imagine *f{x,y)* și vrem să aflăm care dintre două imagini/^xy) și/ 2 (x,>) este o aproximare mai apropiată de/Cx^). Niciuna dintre măsurile convenționale ale diferenței dintre funcțiile Iwo nu oferă rezultate care să fie în acord cu ceea ce ar spune un observator uman. Vezi (2.SA1 pentru o discuție despre acest subiect.

3.7 LITERATURA RELEVANTA

[3.ASI Abramatic. JF și Silverman. L. M „Restaurarea liniară non-staționară a imaginilor zgomotoase". *Proc. 18 Conferința de decizie și control IEEE, vol. I,* **(decembrie 1979). pp. 92-99.**

(3.HA1 Haralick. RM „Statistical and Structural Approaches to Texture." *Proc. Fourth Intern. Joint Conf, on Pattern Recognition.* **(noiembrie 1978). pp. 45-69.**

[3JI) Julesz, B. *Foundations of Cyclopean Perception.* **Chicago: University of Chicago Press. 1971.**

(3J2) Julesz. B.. „Experimente în percepţia vizuală a texturii." *științific american.* **232 (1975). pp. 34-43.**

[3.NM1 Nagao. M. și Matsuyama, T. „Edge Preserving Smoothing". *Proc. Al patrulea intern. Joint Conf, on Pattern Recognition,* **(noiembrie 1978), pp. 518-520.**

(3.OS1 Oppenheim. AV și Schafer, RW *Digital Signal Processing.* **Englewood Cliffs, New Jersey: Prenticc-Hall, 1975.**

[3.PA| Pavlidis. T. *Recunoașterea modelelor structurale.* **Berlin. Heidelberg. New York: Springer Verlag. 1977.**

[3ST1 Steiglitz. K. /În *Introducere în sistemele discrete,* **New York: J. Wiley. 1974.**

3.3 PROBLEME

3.1. Încorporați algoritmul de egalizare a histogramei. Algoritmul 3.2, în programul de supratipărire pe scară de gri al Problemei 1.6.

3.2. Scrieți un program pentru evaluarea matricelor de co-ocurență.

3.3. Se poate defini *dominanța diagonală* pentru o matrice cu elemente nenegative ca raportul dintre suma elementelor sale diagonale și suma elementelor sale diagonale. Ce tip de informație despre aspectul unei imagini transmite dominanța diagonală, D, a unei matrice de co-ocurență? Este posibil ca *D* să fie zero? Infinit? Utilizați programul problemei anterioare pentru a evalua *D* peste părți ale diferitelor imagini.

3.4. Această problemă presupune că ați lucrat la editorul de imagini (Probleme 1.5 și 2.5). Îl puteți extinde acum prin încorporarea în el a procedurilor din Problemele 3.1 și 3.2. Principala problemă care trebuie rezolvată este unde și cum va fi afișată tograma lui sau matricea de co-ocurență.

Sugestii. Aveți două opțiuni: puteți fie să lăsați deoparte o parte a ecranului

pentru afișarea statisticilor zonei vizate, fie să alocați părți din memoria de reîmprospătare pentru stocarea acestor statistici și apoi să utilizați două hărți „color” (vezi instrucțiunea 13 din Tabelul 1.2) pentru a comuta între cele două.

3.5. Proiectați un filtru trece-jos pentru imaginile în care se știe că majoritatea marginilor sunt fie verticale, fie orizontale.

3.6. Implementarea ecuației (3.3) necesită să păstrăm două copii ale imaginii: intrare și ieșire. Să presupunem că în schimb parcurgem imaginea și înlocuim valoarea fiecărui pixel cu suma ponderată a părții din dreapta a ecuației. Aceasta înseamnă că noua valoare va fi utilizată în calculele ulterioare . Investigați efectele unui astfel de filtru.

3.7. Luați în considerare următorul filtru neliniar: pentru fiecare pixel se calculează diferența D_y dintre cei doi vecini de deasupra și dedesubt . precum şi diferenţa D_h dintre vecinii din dreapta şi din stânga. Dacă D_t depășește D*. atunci valoarea pixelului este înlocuită cu media ponderată a lui însuși și a celor doi vecini orizontali ai săi, altfel este înlocuită cu media ponderată a lui însuși și a celor doi vecini verticali ai săi. (Înlocuirea se face în modul ecuației (3.3) și nu ca în problema (3.6).) Investigați efectele filtrului pentru diferite combinații de greutăți. Investigați-le și atunci când filtrul este aplicat în mod repetat pe o imagine.

3.8. Concepeți un gadget pentru fotograful amator care are mai mulți bani decât talent. Aceasta ar trebui să fie o cutie neagră care realizează fotografii supraexpuse sau subexpuse și produce copii cu iluminare corectă. De asemenea, ar trebui să îmbunătățească aspectul imaginilor slab focalizate, să includă o opțiune în care fotograful nu trebuie să specifice ce fel de corecție se dorește. Crezi că poți oferi o opțiune de corectare a fotografiilor când subiectul s-a mutat? Când camera sa mișcat?

Capitolul 4

SEGMENTAREA

4.1 INTRODUCERE

Segmentarea identifică zonele unei imagini care par uniforme pentru un observator și subîmparte imaginea în regiuni cu aspect uniform . Este relativ ușor să definiți uniformitatea în ceea ce privește nivelul de gri sau culoarea. Este mult mai dificil să precizăm ce înțelegem prin „textură uniformă ”.

În general, putem efectua segmentarea în două moduri diferite. Putem presupune că știm dinainte caracteristicile regiunilor sau putem încerca să găsim acele caracteristici în timpul procesării . Ca exemplu simplu, luați în considerare segmentarea după culoare într-o imagine în care fiecăreia dintre cele trei culori de bază (roșu, verde și albastru) îi sunt alocați doi biți în fiecare pixel. Pentru a găsi regiunile roșii ale unei imagini, trebuie să verificăm dacă biții adecvați ai fiecărui pixel sunt egali cu unu în timp ce biții rămași sunt zero.t Pe de altă parte, avem o problemă mai grea dacă ni se cere să împărțim imaginea în două seturi de regiuni, de două culori posibile, fără a primi culorile în prealabil. În acest din urmă caz trebuie fie să studiem statisticile imaginii și să căutăm modalități, fie să examinăm grupuri de pixeli și să verificăm dacă aceștia formează o regiune uniformă. Deoarece subiectul segmentării în întreaga sa generalitate depășește sfera acestui text, ne vom limita

f Soluția sugerată este prea simplă deoarece detectează doar regiuni de culoare roșu aprins. O soluție mai realistă ar trebui să relaxeze aceste condiții.

acoperire la unele dintre cele mai simple scheme. Secțiunea 4.2 examinează pragurile. Secțiunea 4.3 se ocupă de detectarea simplă a marginilor, iar Secțiunea 4.4 prezintă câteva tehnici de creștere a regiunilor.

Analiza scenei este un subiect legat de segmentare. Scopul său este de a oferi descrieri ale imaginilor într-o anumită limbă. Este posibilă separarea celor două procese și efectuarea analizei după finalizarea segmentării . Cu toate acestea, mulți cercetători folosesc informații *a priori* despre scenă pentru a ghida segmentarea. Astfel de tehnici depășesc cu mult procesarea semnalului și algoritmi și, prin urmare, sunt în afara domeniului de aplicare al acestui text.

4.2 PRAGAREA

În această tehnică, valoarea luminozității fiecărui pixel este comparată cu o valoare *de prag* , iar pixelul este atribuit uneia dintre cele două categorii, în funcție de dacă pragul este depășit sau nu. Selectarea valorii de prag se face de obicei din histogramă (discută în Secțiunea 3.2). Într-adevăr, să presupunem că avem de-a face cu imagini de clasa 1 care satisfac ipotezele din Exemplul 3.1, dar pentru care nu avem informații despre forma și locația regiunilor *B* și *C.* Dacă o imagine constă într-adevăr din două regiuni, una predominant luminoasă și cealaltă întunecată, ne putem aștepta ca histograma sa să aibă două vârfuri, așa cum se arată în Figura 3.5 (Figura 3.5). Apoi, un prag *T* poate fi ales dintre valorile dintre cele două vârfuri, iar regiunea *C* poate fi definită ca mulțimea tuturor pixelilor a căror valoare este sub *T.* Figura 4.1 (Plansa 13) arată rezultatul acestei operații pentru originalul din Figura 3.5 pentru *T* = 100. Pragul este cea mai simplă tehnică posibilă pentru transformarea imaginilor în clasa 21, pentru transformarea clasei 21. Din păcate, nu este întotdeauna posibilă selectarea în avans a valorii pragului, deoarece nivelul mediu de luminozitate poate varia și trebuie evaluată histograma pentru fiecare imagine.

În general, evaluarea automată a pragului este o problemă netrivială. Prezența a două vârfuri distincte în histogramă nu este o apariție comună. Figura 4.2a (Plansa 14) prezintă o imagine care pare a fi un bun candidat pentru prag, dar a cărei histogramă (Figura 4.2b (Planca 14)) are un singur vârf major, corespunzător fondului . Obiectele corespund vârfului minor între aproximativ 32 și 64. Figura 4.3 (Plansa 15) prezintă rezultatele pragării pentru *T* = 100. Uneori este de preferat să se evalueze histograma nu peste toți pixelii imaginii, ci doar peste cei care se află în apropierea granițelor - regiunilor. Acest lucru poate fi implementat cu ușurință prin adăugarea unui test în algoritmul 3.1 în care valoarea unui pixel este comparată cu cea a vecinilor săi. Atunci *H(/ (P))* este incrementat numai dacă aceste valori diferă cu

mai mult decât cantitatea de zgomot așteptată. Motivul din spatele acestei restricții este că percepția umană este foarte sensibilă la schimbările nivelurilor de luminozitate și mai puțin la nivelurile absolute.

4.3 DETECȚIA MARCHII

Aceste scheme caută *margini* între regiuni. Ei folosesc un operator *de gradient* , urmat de o operație de prag pe gradient, pentru a decide dacă a fost găsită o margine. Apoi pixelii care au fost identificați ca margini trebuie legați pentru a forma curbe închise în jurul regiunilor. În această secțiune vom discuta.doar câteva dintre cele mai simple detectoare de margine. O tehnică comună este de a lua diferența dintre două grupuri de pixeli în maniera unui filtru liniar trece-înalt de tipul dat de ecuațiile (3.10). Pentru a ține cont de diferențele de orientare a marginilor, trebuie să folosim mai mult de un astfel de filtru. Dacă definim o matrice *H* cu *H^* = *h(ij),* atunci două dintre cele mai simple filtre posibile sunt date de următoarele forme:

^honz	— 1 -*c* -1 0 0 0 1 *c* 1	(4.1a)

*H^* =	eu oi *c 0 — c* 1 0 -1	(4.1b)

Literatura de specialitate conține multe lucrări care descriu filtre cu valori ale *lui c* egale cu 1 sau 2 (vezi Notele Bibliografice). Deoarece filtrele high-pass tind să sporească zgomotul, această clasă de tehnici are o valoare limitată pentru imaginile zgomotoase. Figurile 4.4 și 4.5 (Planșe 16 și 17) arată rezultatele testelor cu detectorul de margine al ecuațiilor (4.1) fore-1 pe originalul din Figura 3.9 (Plansa 10). Rețineți că selectarea unui prag pentru a decide că am găsit o margine este mult mai simplă decât selectarea unui prag pentru segmentare. Un alt exemplu de detectare a marginilor poate fi văzut în Figura 4.6 (Planca 18). Rezultatele de aici sunt mai bune decât cele din Figura 4.5 din cauza contrastului mai mare.

Există scheme mai elaborate de detectare a marginilor care depășesc unele dintre dezavantajele schemelor simple descrise aici doar la un cost de calcul mare. Exemplele prezentate în figurile 4.4 până la 4.6 pot lăsa cititorului impresia că imaginile pentru care detectarea marginilor dă rezultate bune sunt și imagini pentru care se aplică pragurile , dar nu este chiar așa. Este probabil adevărat că detectoarele simple de margine sunt potrivite doar pentru imagini cu contrast ridicat cu puțin înalt

zgomot de frecventa. Cu toate acestea, nu toate astfel de imagini pot fi segmentate corect prin prag. În special, zgomotul *de joasă frecvență* , care nu interferează cu detectarea marginilor, poate face pragurile inutile.

Cel mai simplu exemplu este o imagine în care există un gradient de iluminare, o întâmplare comună în practică. Ochiul uman este sensibil la contrast și schimbările netede ale iluminării pot fi ignorate. Cu toate acestea, ceea ce arată ca „întunecat" la un capăt al imaginii are aceeași valoare ca ceea ce arată ca „luminos" la celălalt capăt, așa că nu există nicio modalitate de a alege un prag pentru separarea zonelor „întunecate" de „luminoase". Pe de altă parte, un simplu detector de margini poate face treaba destul de bine. Această concluzie poate fi valabilă chiar și în cazurile în care zgomotul de joasă frecvență nu este la fel de puternic, deoarece poate fi mai ușor să alegeți pragul pentru a decide prezența unei margini decât pragul pentru segmentarea întregii imagini. Figura 4.7 oferă o ilustrare simplă a acestor puncte.

00000000000000000000000000000000
0000000000333333000000000000000
1111111444444444444411111111111
1111114444444444444111111111111
2222255555552225555555522222222
222225555522222255555555222222222
3336666633333333666666633333333
333666666633333366666666333333
444777777777777777777777777444
447777777777777777777777777444
5588888885555555555588888888855
58888888855555555555555888888885
699999999966666666669999999996
6666666666666666666666666666666
77777777777777777777777777777
77777777777777777777777777777777

Figura 4.7 O ilustrare a avantajelor detecției simple a marginilor față de prag. Cititorii pot detecta probabil forma unei litere din figură și ar putea proiecta, de asemenea, un detector de margini pentru a o găsi automat (vezi problema 4.1).

4.4 SEGMENTARE PE REGIUNE CREȘTERE

În timp ce detectarea marginilor și limitarea se concentrează pe diferența dintre valorile pixelilor, creșterea regiunii caută grupuri de pixeli cu luminozitate similară . În forma sa cea mai simplă, metoda începe cu un pixel și apoi își examinează vecinii pentru a decide dacă au similare.

luminozitatea. Dacă o fac, atunci sunt grupate pentru a forma o regiune. În acest fel, regiunile sunt dezvoltate din pixeli unici. Formele mai avansate nu încep cu pixeli, ci cu o partiție a unei imagini într-un set de regiuni mici. Un test de uniformitate este apoi aplicat fiecărei regiuni, iar dacă testul eșuează regiunea este subdivizată în elemente mai mici. Acest proces se repetă până când toate regiunile sunt uniforme. Apoi regiunile sunt dezvoltate din regiuni mai mici, mai degrabă decât din pixeli. Detaliile implementării depind puternic de structurile de date utilizate pentru reprezentarea unei imagini. Tema creșterii regiunilor va fi discutată în Capitolul 6, ca exemplu de aplicare a unor astfel de structuri. Avantajul major al utilizării regiunilor mici, mai degrabă decât a pixelilor, este o sensibilitate redusă la zgomot.

4.4.1 Segmentarea după nivelul mediu de luminozitate

Un criteriu de uniformitate se bazează pe compararea diferenței maxime dintre valoarea unui pixel și media pe o regiune. Pentru o regiune *R* de dimensiune *N* fie

(42a) ᴬ P,R

Atunci o regiune se numește uniformă dacă

$$\max_{P,R} 1/(P) — m \backslash < T \qquad (4.2b)$$

pentru un anumit prag *T.* Ne putem gândi la această definiție a uniformității ca fiind euristică, dar este posibil să oferim o justificare teoretică pentru aceasta în anumite ipoteze. Următoarea analiză ilustrează multe dintre problemele implicate în creșterea regiunii și poate fi folosită ca model pentru analiza altor criterii de uniformitate.

Vom presupune că imaginile de clasa 1 cu care ne ocupăm sunt de fapt imagini de clasa 2 cu zgomot alb gaussian mediu zero adăugat acestora. Aceasta înseamnă că probabilitatea ca valoarea zgomotului să fie *z* la pixelul *P* va fi dată de următoarea ecuație:

2 ' 2 < 4 ^)

unde *o* este abaterea standard a zgomotului. Valorile lui p„(z) nu depind deloc de locația pixelului *P* , deoarece am presupus că zgomotul este alb În termeni fizici, aceasta înseamnă că zgomotul afectează toți pixelii în *același* MOD .

< **4jb)**

Factorul 2 intră pentru că luăm în considerare atât abaterile negative, cât și pozitive de la medie. Partea dreaptă a ecuației de mai sus este cunoscută ca *funcția de eroare, erf (t),* unde *t* este raportul dintre x și *a.* Majoritatea cărților cu tabele matematice includ liste cu valorile acesteia. Enumerăm câteva dintre ele în tabelul 4.1.

Tabelul 4.1: Funcția de eroare

t	**la**	**15**	**20**	**2.5**	**3.0**	**3 5**	**40**
erf (l)	**0,317**	**0 134**	**0 046**	**0,012**	**0 003**	**0,0005**	**0 0001**

Dacă o regiune este uniformă, atunci ecuația (4.2a) este estimatorul optim al valorii luminozității acolo (vezi problema 4.4). Într-un astfel de caz, abaterile valorilor pixelilor de la *m* se vor datora numai zgomotului și, prin urmare, probabilitatea ca ecuația (4.2b) să nu fie valabilă pentru un pixel va fi dată de ecuația (4.3b) dacă alegem z - 7, în mod specific, va fi egal cu *erf(T/a).* De exemplu, dacă alegem 7 în ecuația (4.2b) egal cu 2*. atunci probabilitatea ca această condiție să eșueze pentru un anumit pixel este de 4,6 la sută, în timp ce pentru 7 — 3a este de aproximativ 0,3 la sută. Vom folosi *p(T)* pentru a desemna valoarea acestei probabilități. Probabilitatea de a satisface ecuația (4.2b) este I — *p(T)* pe pixel și, deoarece *N* este numărul de pixeli pe regiune, atunci nu vom recunoaște o regiune uniformă doar cu probabilitatea I - (I — p(7) 1*. Pentru valorile *p(T)* mult mai mici decât —, această cantitate este egală cu aproximativ N *p(T).* presupunând o regiune pătrată de 16X16 de 256 de pixeli, rezultă o probabilitate de eșec pentru testul de uniformitate egală cu 54 % (folosind formula exactă, dacă pragul este de patru ori deviația standard, atunci aceeași probabilitate este de numai 2,5% , o valoare mai acceptabilă din punct de vedere practic, dar și formula exactă și similară.2). 0,0256 respectiv).

Numirea neuniformă a unei regiuni uniforme nu este singura eroare posibilă și acum trebuie să estimăm probabilitatea ca o regiune neuniformă să fie numită uniformă. În acest caz, diferențele dintre *m* și valorile pixelilor se vor datora și diferențelor dintre valorile regiunilor imaginii de clasa 2. Fie m și m_2 aceste valori și fie *q*> procente din pixelii unei regiuni să fie pixeli a căror valoare adevărată este *m, {i* " 1.2). Dacă regiunea este suficient de mare pentru a neglija efectul zgomotului în estimarea mediei, atunci media va fi <7rm, + $q_2 nt_2$. Dacă

un pixel are valoarea adevărată $w_{|t,}$ atunci diferența dintre această valoare și media estimată va fi

$$6m - m_{\{} - (fl / mi + q\^m\^ . \qquad (4.4)$$

În consecință, evenimentul în care valoarea sa observată diferă cu mai mult de *T* față de $q\^mt + q_2 m_2$ poate avea loc prin faptul că acesta diferă de $m_($ fie cu *T + 6m,* fie cu *T - 6m* și probabilitatea ca oricare dintre evenimente să se producă este egală cu

$$Pi ""I H*^{7} " \^I) + 7(|T + «ml)] \blacksquare \qquad (4,5)$$

Prin urmare, p_x este probabilitatea ca valoarea observată a unui pixel cu valoarea adevărată $m_{\{}$ să încalce ecuația (4.2b). Probabilitatea ca niciunul dintre pixeli să nu documenteze acest lucru este

P. „(lp.)"" «-^)”" (4.6)

unde p_2 este definit într-o manieră similară cu p_x. Astfel p, este probabilitatea de a accepta o regiune ca uniformă atunci când de fapt nu este. În mod clar, dacă *6m* este mic în comparație cu *T,* atunci pj este aproape de *p(T).* Dacă acest lucru este valabil și pentru p_2, atunci *p„* este aproximativ [1 - *p(T)]* ^{N}, care este aceeași cu probabilitatea de a numi uniformă o regiune cu adevărat uniformă. Cu alte cuvinte, detectarea uniformității va părea un eveniment aleatoriu. Situația în care *6m* este mic este probabil să apară atunci când o regiune conține pixeli de aproape un singur tip, deci eroarea nu este o regiune similară amestec egal de două tipuri de pixeli, adică dacă fl, " q_2^ astfel încât

«m=y^i]-m $_2$], (4,7)

atunci am dori să se numească uniformă cu o probabilitate foarte mică. Acest lucru se va întâmpla dacă se așteaptă ca diferența *de 6m* să fie mult mai mare în valoare absolută decât deviația standard a zgomotului, deoarece *6m* va fi atunci de dimensiune comparabilă cu *T.* În aceste condiții, argumentul primului termen din suma ecuației (4.5) va fi aproape de zero și, prin urmare, probabilitatea corespunzătoare va fi aproape de 1. Al doilea termen va avea un argument egal cu o deviație standard, atât de aproape de zero, încât probabilitatea standard va fi atât de aproape de un multiplu de zero. Aceste ipoteze produc pi egal cu aproximativ 0,5 și aceeași valoare poate fi găsită pentru p_2. Atunci *p„ va fi aproximativ* egal cu 0 5* În acest caz , încălcarea ecuației (4.2b) implică o regiune neuniformă cu probabilitate 1 — 0,5*.

4.4.2 Alte criterii de uniformitate

Analiza anterioară arată că criteriul de uniformitate al ecuației (4.2b) este probabil să greșească pe partea de a numi o regiune neuniformă prea des. Acest lucru poate explica, cel puțin parțial, o experiență comună atunci când o imagine este segmentată folosind un criteriu similar cu ecuația (4.2b): există un număr mare de regiuni mici care nu par să aibă omoloage reale în imagine. O analiză completă a altor criterii depășește sfera acestui text, la fel ca și o discuție a efectelor mărimii regiunii *N* asupra fiabilității estimărilor pentru media regiunii (vezi Notele bibliografice). O definiție oarecum diferită a uniformității poate fi făcută prin compararea statisticilor dintr-o regiune cu statisticile evaluate pe părți ale acesteia. Dacă sunt aproape unul de celălalt, atunci regiunea poate fi numită uniformă. O astfel de abordare poate fi utilă pentru segmentarea după *textură.* Se poate evalua matricea de co-ocurență pe fiecare dintre un grup de regiuni și apoi se poate compara matricele. Dacă sunt similare, unirea acestor regiuni este o regiune uniformă. În general, fie *F(R)* o *caracteristică* estimată pe regiunea *R.* Dacă /? $_{l2}$ este uniunea a două.regiuni adiacente dar disjunse *R^* și *R* $_2$, atunci s-ar putea defini un criteriu de uniformitate cerând ca *F(R* $_{l2}$*)* să fie aproape de *F(Ri)* și *F(R* $_2$*)- R* ar putea fi regiunea deja găsită și f? $_2$ o regiune mică care este considerată pentru adăugare la *R^* (a se vedea capitolul 6 pentru alte interpretări). Trebuie să alegem un prag *T'* astfel încât atunci când valoarea absolută a diferenței *F(Ri)* și *F(R* $_2$*)* este sub *T',* să decidem în favoarea uniformității. O analiză similară cu cea din Secțiunea 4.4.1 arată că *T'* trebuie să fie mai mare decât varianța lui *F(R)* cauzată de zgomot. Cu toate acestea, în multe cazuri, această variație este mult mai mică decât varianța zgomotului în sine, în timp ce varianța datorată neuniformității este de aceeași dimensiune ca înainte. Astfel, un criteriu de uniformitate bazat pe compararea caracteristicilor este mai fiabil decât cel dat de ecuația (4.2b). (Desigur, mijlocul în sine este o astfel de caracteristică.)

4.5 NOTE BIBLIOGRAFICE

Tehnicile avansate de segmentare sunt de obicei tratate în contextul - recunoașterii modelelor. Consultați capitolele 4-6 din [3.PA] și capitolele 7 și 8 din [2.HA] pentru detectoare de margine mai elaborate, tehnici de creștere a regiunii și analiză a scenei. Lucrarea lui Abdou și Pratt (4.AP] oferă o analiză detaliată și o comparație a schemelor mai simple de detectare a muchiilor, similare cu cele date de ecuația (4.1). Brooks [4.BR] oferă o justificare teoretică pentru diverși detectoare de margini. Vezi [4.CP] pentru un tratament al segmentării ca problemă de estimare și o discuție a efectelor fiabilității regiunii, a dimensiunii estimărilor etc.

Sperăm că analiza secțiunii 4.4 a demonstrat necesitatea modelelor de imagine. Acestea nu trebuie confundate cu modelele de obiecte utilizate în analiza scenei, deși pot fi adesea legate de acestea. Modelele de analiză a scenei tind să fie deterministe, în timp ce modelele de imagine sunt în primul rând stocastice și au de-a face cu distribuția valorilor așteptate ale zgomotului și diferențele în statisticile regiunii. Volumul 12 (1980) al revistei *CGIP* este dedicat subiectului modelării imaginilor. [4.CE], (4.DM] și [4.HA] sunt unele dintre lucrările de acolo care sunt deosebit de relevante pentru tema segmentării.

[4.BA] este o revizuire recentă a analizei scenei, cu accent pe modele tridimensionale și modele de iluminare. Acesta este un subiect în care interacțiunea dintre procesarea imaginii și metodologiile grafice poate fi foarte fructuoasă, dar munca în acest domeniu este abia în primele etape.

4.6 LITERATURA RELEVANTA

[4.API Abdou, IE și Pratt, WK „Proiectare și evaluare cantitativă of Enhancement / Thresholding Edge Detectors," *IEEE Proceedings,* **67 (1979), pp. 753-763.**

14.BA] Bajcsy, R. „Three-dimensional Scene Analysis", *Proc. Al cincilea intern. Conf, pe Pattern Recognition.* **Miami Beach, decembrie 1980, p. 1064-1074. (Publicat de IEEE Computer Society, Catalog IEEE Nr. 80CH1499-3.)**

[4.BR] Brooks, MJ „Rationalizing Edge Detectors", *CGIP.* **8 (1978). p. 277-285.**

[4.CE] Cooper, DB; Elliott, H.; Cohen, F.; Reiss, L.; și Symoser, P. „Stochastic Boundary Estimation and Object Recognition", *CGIP.* **12 (1980), p. 326-356.**

[4.CP] Chen, PC și Pavlidis, T. „Image Segmentation as an Estimation Problem", *CGIP.* **12 (1980), p. 153-172.**

1 4.DM] Davis, LS și Mitiche, A. „Edge Detection in Textures", *CGIP.* **12 (1980). pp. 25-39.**

14 .HA] Haralick, RM „Edge and Region Analysis for Digital Image Data", *CGIP.* **12 (1980), p. 60-73.**

15 7 PROBLEME

4.1. Proiectați un detector de margini pentru segmentarea imaginii din Figura 4.7.

4.2. Localizarea unei margini pe o imagine poate fi tratată ca o problemă pentru găsirea regiunilor neuniforme. Încercați să modificați analiza secțiunii 4.4.1 pentru studierea detectorilor de margine date de ecuația (4.1).

4.3. O posibilă alternativă la ecuația (4.2b) este compararea valorii erorii pătratice medii cu un prag (în loc de valoarea erorii maxime). Crezi că aceasta este o abordare rezonabilă?

4.4. Demonstrați că valoarea lui *m* dată de ecuația (4.2a) minimizează eroarea pătrată integrală dintre valorile lui *f(P)* și *m. (Notă:* dovada este conținută în majoritatea textelor standard despre statistică sau procesarea semnalului, dar poate dura mai puțin timp pentru a o obține decât căutarea într-o carte.)

4.5. Utilizați editorul pentru a efectua o evaluare interactivă a statisticilor imaginilor. Poziționați o fereastră peste o zonă care vi se pare uniformă și apoi apelați o procedură de evaluare a statisticilor. (Histograma este unul dintre ele.)

Capitolul 5

PROIECȚII

5.1 INTRODUCERE

În procesarea imaginilor, termenul *de proiecție* se referă de obicei la maparea unei imagini într-o formă de undă ale cărei valori sunt sumele valorilor punctelor imaginii de-a lungul anumitor direcții. (În grafică, precum și în unele probleme de analiză a scenei, termenul este folosit în mod obișnuit pentru a desemna maparea unui obiect tridimensional la plan cu toate informațiile de adâncime pierdute.) *Reconstituirea* imaginilor în scala de gri din proiecțiile lor este una dintre cele mai cunoscute utilizări ale computerelor pentru aplicații medicale, deoarece oferă un instrument puternic de diagnosticare care înlocuiește unele tehnici dureroase și periculoase (vezi Notele bibliografice). Tehnica poate fi extinsă pentru a reconstrui un obiect tridimensional din proiecțiile sale bidimensionale. Acest lucru se realizează prin legarea împreună a unei serii de secțiuni transversale. De obicei, obiectul este un organ uman (inima, creierul etc.) iar proiecțiile sunt imagini radiografice sau ultrasonice, dar teoria este aplicabilă oricărui tip de obiect și semnal penetrant. Figura 5.1 (Plansa 19) ilustrează rezultatele unei astfel de reconstrucții. Vom discuta aici câteva dintre tehnicile mai simple, fără a intra în nicio discuție despre aplicațiile medicale. Secțiunile 5.2 și 5.3 descriu o metodă fundamentală pentru astfel de reconstrucții.

Proiecțiile au fost, de asemenea, utilizate pentru *analiza formei* și această -aplicație este discutată în Secțiunea 5.4. În cele din urmă, Anexa este o listă a unui program de reconstrucție implementat pe un computer de uz general utilizând supratipărirea în scară de gri pentru rezultat.

5.2 INTRODUCERE ÎN TEHNICILE DE RECONSTRUCȚIE

Dacă *d(x ^,z)* este densitatea unui obiect în punctul *(x^.z),* iar dacă absorbția razelor X este proporțională cu densitatea cu un factor *k,* atunci o radiografie (negativă)

paralelă cu planul (yz) va avea o funcție de luminozitate:

p{y,z)-k^f(xy!-)dx (5.1)

unde L este o linie care trece prin obiect.

Dacă razele X nu sunt de-a lungul axei X, ci formează un unghi 0 cu aceasta (secțiunea Figura 5.2), atunci integralele vor fi calculate de-a lungul liniilor cu ecuații

xsin^-ycos^—f (5.2)

unde *I* este distanța euclidiană a unei linii de la origine. Atunci avem

p{4,t .z^kfd{xj,zMxsind>-ycos$-t]dxdy (5.3)

▶

unde v este un volum care înconjoară obiectul și 61) denotă o funcție delta.! Introducerea funcției delta ne permite să definim linia de integrare implicit, mai degrabă decât explicit, ca în ecuația (5.1).

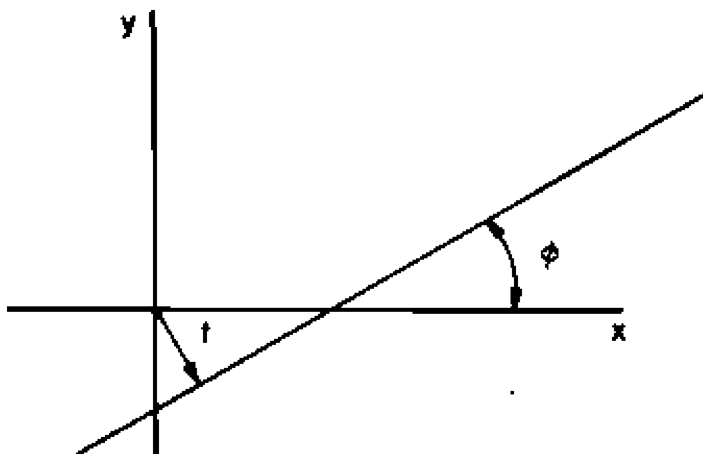


Figura 5.2 Definiția cantităților $ și *t* utilizate în ecuația (5.2)

t Thu este o funcție care este peste tot zero, cu excepția cazului în care argumentul său u zero și a cărei integrală de la -co la +co este egală cu unu.

În practică, proiecțiile sunt Uken numai pentru o valoare fixă de *z,* iar o *secțiune transversală* a obiectului este reconstruită și afișată. (Desigur, repetarea procesului pentru *z diferit* va produce informații despre întregul obiect.) Apoi putem renunța la dependența de *z* din ecuația (5.3) și, notând cu *f (xj)* absorbția la (x,>). noi obuin

P(^^)"^/(xj»)*|xsin4-ycos4-f)4td> . (5.4)

R este o zonă care conține secțiunea transversală de interes. Deoarece presupunem /(*J')"0 în afara obiectului, forma exactă a lui *R* nu este importantă.

Reconstituirea lui */(xy)* dintr-o mulțime de *p(4>,l)* pentru diferite valori ale lui 0 se bazează pe un rezultat matematic care arată că transformata Fourier bidimensională a lui *f (xj)* poate fi găsită din mulțimea transformărilor Fourier unidimensionale ale lui *p(4>,t).* Atunci *f {xy)* poate fi găsit luând o transformare inversă. Acest rezultat poate fi dovedit destul de ușor după cum urmează.

Fie P(^,w) *FT al lui p($,t)* dat de ecuația:

Plo.w)- e-^piMdi . (5,5)
— co

Înlocuind ecuația (5.4) în ecuația (5.5) găsim

P(0.w)-J''' ^*/(x^e-^^-'^'dxdy ,* (5.6)
— co — co

unde am folosit faptul că integrandul este zero dacă nu r—xsin^—ycos^. O comparație cu ecuația (2.2) arată că aceasta este transformata Fourier bidimensională a lui *f (x ,y)* calculată la frecvențele wsin^ și —wcos^. Se poate demonstra că transformata Fourier inversă pentru cazul bidimensional este dată de o ecuație analogă cu ecuația (2.6), și în special

/(xj-)--^ j“ *J^F^.^eK^dudv* (5.7)
4r —QO -co

P(<t>,w) este de fapt F(u,v) în coordonate polare cu uwin și v = —wcosd-Transformarea din coordonate carteziene în coordonate polare înlocuiește diferența *dudv* cu M^'rfd unde |w| este determinantul matricei iacobiene a transformării. Astfel găsim

*f(*J>* $= \frac{1}{4\pi^2}\int_0^{\pi} d\phi \int_{-\infty}^{\infty}$ /'(^I*)e/-(,‘,M'X<O,"I*I^ (5.8)

Ecuații (5.4). (5.5) și (5.8) rezumă fundamentul matematic al reconstrucției prin proiecții. Teoretic, este posibil pentru a realiza o reconstrucție „perfectă” calculând *p(4>,t)* pentru un număr suficient de mare de unghiuri. Cu toate acestea, există multe dificultăți practice înainte de a discuta aceste ecuații, deoarece aceasta este ceea ce este implementată în practică, deoarece ecuația de proiecție este întotdeauna analogă

PiW-Z' P<tk) wO.!. • • ■ WI (5,9) *-0

unde *N* este numărul de puncte în care a fost cuantificată o dreaptă perpendiculară pe

direcția de proiecție. Ecuația (5.7) devine

/(*./)--4-S £ F(uv)e '' (5.10)

U-0 »-0

Ecuația (5.9) furnizează valorile lui F(u,v) sub forma Ffwsin^.-wcos^). Rețineți că în acest fel nu putem obține / pentru exact *N* } valori egal distanțate ale argumentului său. Acest lucru este ilustrat în Figura 5.3. Avem mult mai multe valori la frecvențe joase decât la înalte. În cazul continuu (Ecuația (5 8)), această lipsă de frecvențe înalte este compensată de factorul Iwl și, prin urmare, este recomandabil să se folosească o versiune discretă a ecuației (5.8) mai degrabă decât a ecuației (5.10). O mare parte din cercetarea în algoritmi de reconstrucție se ocupă în esență de compensarea adecvată pentru distribuția neuniformă a valorilor disponibile ale transformării Fourier.

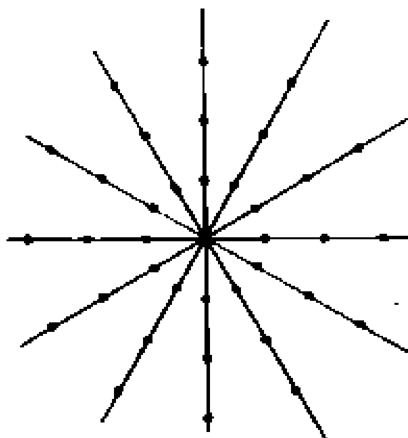


Figura 5.3 Distribuția valorilor disponibile ale transformării Fourier. Probele (cercurile pline) sunt distanțate uniform de-a lungul razelor, dar distribuția nu este uniformă în sistemul de coordonate *UV* **.**

5.3 O CLASĂ DE ALGORITMI DE RECONSTRUCȚIE

Problemele practice majore în implementarea reconstrucției sunt necesitatea unui calcul rapid, astfel încât rezultatele să poată fi afișate în timp real, sau aproape în timp real, și nevoia de a menține zgomotul și distorsiunea la un nivel scăzut. Vom descrie o metodă populară de reconstrucție, cea a *convoluției .*

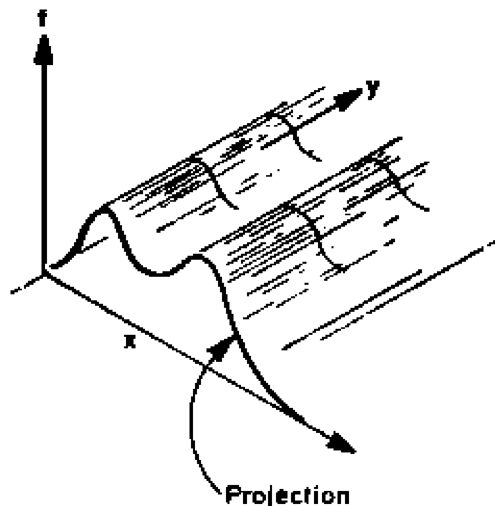


Figura 5.4 Definirea retroproiecției. Curba în planul / -x este translatată de-a lungul direcției lui *y* axa pentru a crea o suprafață cilindrică. Suprafața, privită ca o funcție a două variabile . x și *y,* este retroproiecția.

Fie integrala interioară a ecuației (5.8) definită ca

?(«.<) „ *P(-i>,w)\w\e^dw* (5.12)

unde

r = xsin0—^0050 .

Apoi

/(xj) '' y^-j\(0.xsind>->cosd>)j0 . (5.13)

Pentru un *<t> fix.* ç(0,xsin^-ycos^) poate fi interpretat ca o *retroproiecție .* adică, o funcție a două variabile definite dintr-o funcție a unei singure variabile, așa cum se arată în Figura 5.4. Atunci reconstrucția este jumătate din

media retroproiecțiilor integrate peste 0. (Factorul 1/2 intră pentru că partea dreaptă a ecuației (5.13) este împărțită la 2», în timp ce integrala este doar de la 0 la r.) Funcția *q{<t>,t)* este inversul *FT* a lui / > (^ 1 M')k|, sau convoluția *lui p(<t,lsert)* k (<t>,t> , *t)* k | adică

q(M)-J™ p(4>.T)r(tT)dt. (5,14)

-co

Din teoria sistemelor liniare, știm că aceasta este ieșirea unui filtru cu răspuns la impuls r(O când intrarea sa este *p(<t>.t). (*^ acționează ca un parametru.) Aceasta conduce la algoritmul de reconstrucție 5.1 unde *M* este numărul de unghiuri 0 măsurate, iar *f r {x^)* este funcția reconstruită.

Algoritmul 5.1 Algoritmul generic de reconstrucție

proiecții *p(4J) ; q(^.t)* proiecții filtrate; *f,(xj)* reconstrucție ; *M* numărul de unghiuri utilizate la calcularea proiecțiilor.

1. Inițializați/,(* O')" 0.
2. **Pentru** fiecare 0 măsurat, **faceți:**
   ÎNCEPE.
3. Treceți *p{d>,t)* prin filtru pentru a găsi *q(d>,t).*
4. **Pentru fiecare (x^) faceți:**
   ÎNCEPE.
5. *fAxj) -frtxj)* + g(d^sin«-ycos0).
   Sfârşit.
   Sfârşit.
6. **Pentru** fiecare (xj^) **faceți:**
   ÎNCEPE.

   Sfârşit.

8 . **Sfârșitul algoritmului.**

Ceea ce rămâne, acum, este alegerea filtrului și aici stau dificultățile practice. Nu există un filtru realizabil fizic cu funcție de transfer kl care să nu amplifice și zgomotul. Se poate spune că pasul critic într-o tehnică de reconstrucție este alegerea lui *r(t).* Următoarele sunt câteva opțiuni posibile:

(a) *r(t*) este funcția delta și funcția de transfer este 1 mai degrabă decât kl. În acest caz nu este nevoie de convoluție deoarece *qi^d^P^J).*

t Folosim un simbol diferit pentru a sublinia că este o funcție calculată mai degrabă decât absorbția adevărată *f (x^).*

Se găsesc pur și simplu retroproiecții din proiecțiile *p(1>,t)* și se calculează media lor ca reconstrucție. Desigur, aceasta va fi distorsionată și distorsiunea poate fi estimată din raportul dintre funcția de transfer a filtrului real și funcția de transfer a filtrului ideal. În acest caz acest raport este de 1/kl, adică un filtru trece jos, astfel încât reconstrucția va fi neclară.

(b) Dacă *a* este distanța dintre razele paralele ale proiecției, alegeți:

r(0) - 4/(<g) ^{2} (5.15a)

r(*a) „-4/(r ^{2} a ^{2} (4**-l)) *-±l,±2, • • • și r(O să fie liniar între aceste (5.15b)

puncte. Acest lucru produce

*k) 2 . ^a — sin-— *a* 2 sin(wg/2) / wg/2 (5.16)

Dacă kl este mult mai mic decât 1/g, atunci

***(w)~kl .** (5.17)

Prin urmare, reconstrucția $f_r(xy)$ va fi exactă dacă F(u,v) este zero pentru |u I și |v| mai mare decât o mică fracțiune de l/g. În acest caz avem și o simplificare a calculului. Într-adevăr, algoritmul este echivalent cu evaluarea formulei

/.(^“^S 5 />(*#.^(xsin^-TCOS-^-*o) . (5.18) y—o *k* — -*N*

Aceasta presupune cuantificare uniformă a unghiului și distanței. *N* depinde de dimensiunea zonei care conține secțiunea transversală.

Există multe variante ale acestor tehnici a căror acoperire depășește domeniul de aplicare al acestui text. În schimb, vom da un exemplu simplu al procesului de reconstrucție.

Exemplul 5.1: Fie secțiunea transversală un cerc cu raza *R* și densitatea unu. Atunci proiecția sa la orice unghi 0 va fi

p(<t> $_t$ I) = iVR ^{2} ^ pentru t ^{2} ^R ^{2} , (5.19a)

p(^J) = 0 în caz contrar. (5.19b)

Rețineți că p(0,r) este de fapt independent de ^. Reconstrucția din două retroproiecții la 90’ fără filtrare conduce la rezultatul exprimat prin următoarele ecuații, unde regiunile de valabilitate ale fiecărei expresii sunt prezentate în Figura 5.5.

(Reg. I) /r(xj>) " (5.20a)

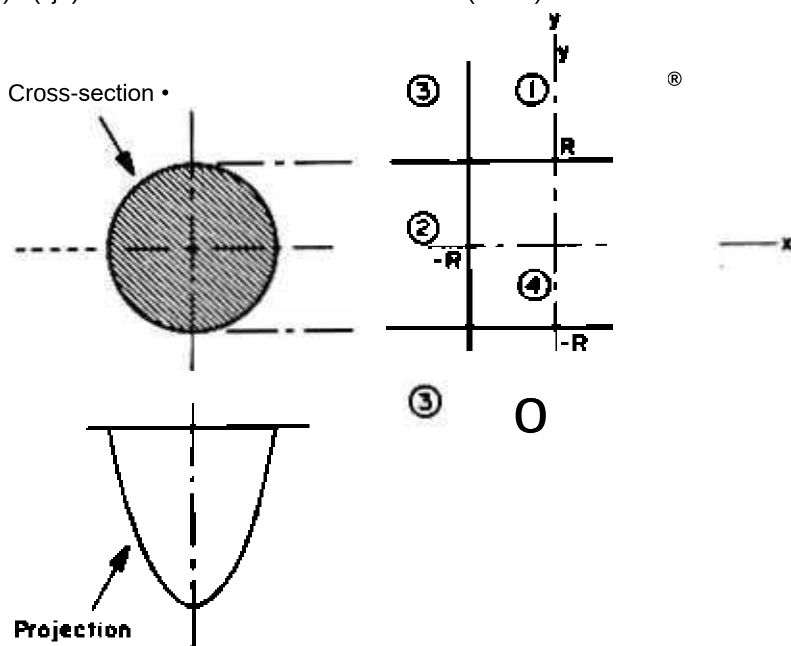


Figure 5.5 A circular cross-section (upper left) has for all angles a projection of the form shown at the lower left. The backprojec* lion of the projection parallel to the x-axis is nonzero for Regions 2 and 4 shown at the right pan of the figure. Similarly, the back- projection of the projection parallel to the /-axis is nonzero for Regions I and 3.

|x|<R and |y|>R |x|>/? and |y|</? |x|>K and 1/1>/? lxl<R and l/l</?

(Reg. 2) AUjJ-x/F? (5.20b)
(Reg. 3) ___________ A(^EL=0 , (5 20c)
(Reg. 4) / r (x^) - (V^-P+V^-P) (5,20d)

Factorul 2 al ecuației (5.19a) a fost eliminat deoarece luăm media celor două retroproiecții. De-a lungul perimetrului cercului (P + y' - R 2) un calcul simplu arată că dacă

x-/?cos^ și /-/?sin^

apoi rezultă ecuația (5.20d).

/,(x jO ~ ^(Isin^l + Icos^l) (5.21)

Pentru 0<^<y avem

fr(xj) - /?(sinf+co$f) = *V2R* cos(^--7) , (5.22)

în timp ce valoarea corectă este constantă și egală cu unu. Este rezonabil să se

normalizeze fiecare proiecție prin împărțire cu aria *A* găsită ca integrală a proiecției în raport cu r. În cazul de față avem

4 - *Zf^VR^di) + 1R* ^{2}. (5,23)

astfel încât reconstrucţia devine

f,(xy) „ -nj-cosW—7-) • (524)

Graficul ecuației (5.24) este prezentat în Figura 5.6.

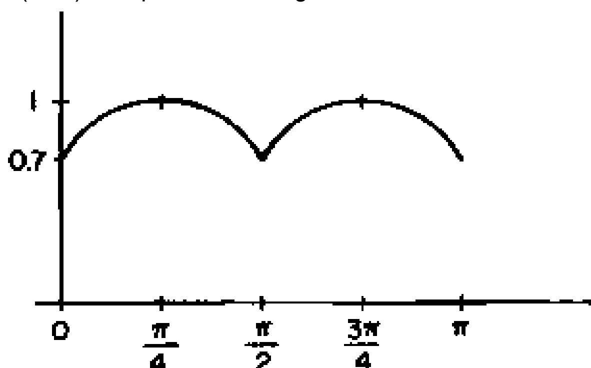


Figura 5.6 **Forma funcției reconstruite de-a lungul periferiei unei regiuni circulare folosind doar două proiecții. Axa verticală este marcată în multipli de** *„J2/*R.*

Următoarele sunt câteva alte valori normalizate *ale f* $_{r}$ *(x,y):*

/(0,0) - 4 *τR '* (5.25a)

Z (0^) = -^*y{R^R'~y* ^{i} -> dacă I/I<* , *rR* ^{2} (5.25b)

/(O^)- dacă bd»? . *x R* (5.25c)

Adăugarea mai multor proiecții va crește valorile *lui f* $_{r}$ *(x,y)* în zona cercului, dar va păstra și valori diferite de zero în afara acestuia, de unde neclaritatea. De exemplu, adăugarea a încă două proiecții va schimba valoarea dată de ecuația (5.25a) la dar va lăsa neschimbată valoarea dată de ecuația (5.25c). □

9 .4 PROIECȚII PENTRU ANALIZA FORMEI

În plus față de utilizarea lor pentru reconstrucție, proiecțiile pot fi utilizate pentru analiza formei, deoarece mapează o regiune bidimensională într-o formă de undă. De exemplu, caracterele alfanumerice sunt scrise cu linii de-a lungul câtorva direcții: verticală, orizontală și două diagonale. Astfel de lovituri vor apărea ca vârfuri în proiecție de-a lungul direcției lor și ar putea fi detectate cu ușurință prin orice număr de tehnici. În această aplicație este practic să folosim doar câteva proiecții și trebuie să calculăm,

de asemenea, ecuația {5.4) în mod explicit, mai degrabă decât să măsurăm partea stângă ca rezultat al unui dispozitiv fizic. Continuăm cu analiza prin rescrierea ecuației (5.4) pentru <*t*> — 0*. 90', 45' și 135*.

p(O',f) - ^/(x./Jà (5.26a)

P(90*.l)-j/^W (5 26b)

p(45*,f) - *f R f{x^~Vll)dx* (5.26c)

p(135'.f)- ^/(x.^-x)^ (5.26d)

Chiar dacă patru proiecții sunt insuficiente pentru o reconstrucție bună, ele oferă informații considerabile despre accidente vasculare cerebrale. O problemă în aplicarea lor este găsirea formei discrete adecvate pentru ecuațiile (5.26c) și (5.26d). Algoritmul 5.2 oferă o implementare simplă care asigură că toate proiecțiile sunt vectori cu același număr de componente și că dimensiunea grilei *N* este pară.

Operațiile efectuate la pașii 6 la 13 au ca efect gruparea diagonalelor 2/Vl ale grilei *NXN* în *N* perechi, fiecare constând din două diagonale, cu excepția celor două din mijloc, care împart diagonala principală. Figura 5.7 (Plansa 20) și Figura 5.8 (Planca 21) prezintă câteva exemple.

Exemplele arată că într-adevăr, informații considerabile despre aspectul brut al personajului sunt conținute în proiecții. Cu toate acestea, acest lucru nu pare să fie cazul cu conturul mașinii. Detectarea vârfurilor în proiecții poate fi realizată prin orice tehnică adecvată în analiza formei de undă. De obicei, nivelul zgomotului este scăzut și sunt aplicabile tehnici simple, cum ar fi pragurile sau chiar diferențierea. Metoda a fost utilizată pentru recunoașterea caracterelor alfabetice cu succes rezonabil, chiar dacă au fost calculate doar două proiecții (vezi Notele bibliografice).

Algoritmul 5.2 Calcularea proiecțiilor pentru analiza formei

Notație: f (i J) **matrice de intrare de dimensiune** *NXN; p(kl)* **matrice de proiecție de dimensiunea 4XJV.**

*-i

1. Pentru/-OtoN-l selp(0,/)- J /(/,/).

io N-

2. Pentru / — 0 la/V—I setp(U)- 2 /(/.»)• *io*
3. Inițializați *p* (2,*) și p(3,*) la zero.
4. Pentru > - 0 la /Vl faceți:

   ÎNCEPE.

5. Pentru / — 0 la / V— eu fac:

   ÎNCEPE.

6. Comparați r+y cu /V—I.
7. Dacă este mai mare, atunci adăugați *f (jj)* top(2,(/+/+l)/2).
8. Dacă este mai mic, atunci adăugați */{JJ)* la *p(2.(i+j)/2).*
9. Dacă este egal, atunci se adaugă *f(jj)/2* la p(2,(W2)-l) și /OJ)/2 top(2.N/2).
10. Comparați *j—i* cu 0.

11. Dacă este mai mare, atunci adăugați */(jj) la p{3,(j—i+N)/2).*

12. Dacă este mai mic, atunci se adaugă *f(j,i) top(3,(J-i+N-\)/2) .*

13. Dacă este egal, atunci adăugați *f(J,i)/2* la p(3,(/V/2)-l) și *f(jj)/2 sus(3,N/2).*

Sfârşit.

Sfârşit.

14. Sfârșitul algoritmului.

Pentru a vizualiza cantitatea de informații prezente în aceste proiecții, încercăm să reconstruim contururile obiectului prin retroproiecții filtrate. Rezultatele sunt prezentate în Figura 5.9 (Planca 22) și Figura 5.10 (Planca 23). Algoritmul de reconstrucție este similar cu Algo ritmul 5.2 și este dat ca algoritmul 5.3.

Exemplele de reconstrucție din figurile 5.7 până la 5.10 sugerează unele dintre puterile și limitările metodei. Detaliile se pierd atunci când conturul este mai degrabă circular decât rectiliniu, ca în cazul conturului mașinii. Cu toate acestea, din cauza acestei lipse de sensibilitate, metoda poate fi adecvată pentru recunoașterea caracterelor multifon. Într-adevăr, proiecțiile pot fi utilizate pentru a detecta lovituri de-a lungul unor direcții specifice. Următorul exemplu prezintă o analiză care poate fi utilă pentru a decide câte proiecții să fie utilizate în aplicații specifice.

Algoritmul 5.3 Reconstrucție prin Backproiecții

Notație: f(ij) matrice de ieșire de dimensiunea *NM-. matrice de proiecție p(kj)* de dimensiunea 4XA.

1. **Pentru** y — 0 la /V — eu **fac: Începe.**
2. **Pentru / — 0 la /V—1 faceți:**

ÎNCEPE.

3. *gUJ)-P^JnpO-O*
4. Comparați *i+j* cu *N—}*
5. **Dacă** este mai mare, **atunci** adăugați p(2,0+y+l)/2) la g(yf)
6. **Dacă** este mai mic, **adăugați** p(2,(f+y)/2) la gQ.i)
7. **Dacă** este egal, **atunci** adăugați 1/2 [p(2,(/V/2)-l)+p(2,N/2)] la *gljj)*
8. Comparați *j—i* cu 0.
9. Dacă este mai mare, **atunci** adăugați *p(3,(J—i+N)/2)* la *g(JJ)*
10. Dacă este mai mic, atunci adăugați *p(3,(j—i+N—l)/2)* la *g(jj)*

H. **Dacă** este egal, adăugați 1/2lp(3,</V/2)-])+p(3./V/2)] la *g(Jj)*

Sfârşit.

Sfârşit.

12. **Sfârșitul algoritmului.**

Exemplul 5.2: Să presupunem că ne interesează doar dreptunghiuri de anumite - dimensiuni și dorim să decidem câte proiecții ar trebui să folosim pentru a le detecta. Poziția unui dreptunghi se definește în raport cu direcția unei proiecții prin ^, unghiul

format între cea mai mare latură a dreptunghiului și direcția perpendiculară pe proiecție (Figura 5.11a). Pentru ^ —0 proiecția este egală cu w pe un interval egal cu *H,* în timp ce pentru ^ " 90' este egală cu *H* pe un interval egal cu *w.* Pentru alte valori ale lui < proiecția are un maxim izolat ,_p „ . *Valoarea* maximă a lui *p* M (față de ^) apare atunci când direcția proiecției este paralelă cu o diagonală *a* reclinării · V***+H*. Acest lucru se realizează pentru ^^ - Un" 1 (///w). Funcția de proiecție la acel unghi are o formă triunghiulară și este nenulă într-un interval egal cu wsin^ + Wcos^, sau 2H>v/p^. Pentru ^ < ^„»x' *Pm* este w/cos^, în timp ce pentru ^m« < ^ < 90', p m . este metrică cu funcția Zf la sin^m^, cu funcția Zf la sin/m^. astfel încât graficul său are forma prezentată în Figura 5.11b În mod clar, dacă folosim o singură proiecție, este posibil să nu detectăm dreptunghiuri când ^ este aproape de 90* Dacă folosim patru proiecții (45* depărtate), atunci în cel mai nefavorabil caz maximul detectat

va fi w/cos(67,5') — 2,41w dacă 67,5' < ^„u, și H/sin(67,5") — 1,087/ în caz contrar. Dacă *H/w* — 5. atunci ^ M1 " 78,7', iar vârful maxim eliminat în poziţia cea mai nefavorabilă va fi aproximativ jumătate din dimensiunea aşteptată. Figura 5.11c ilustrează această poziţie. □

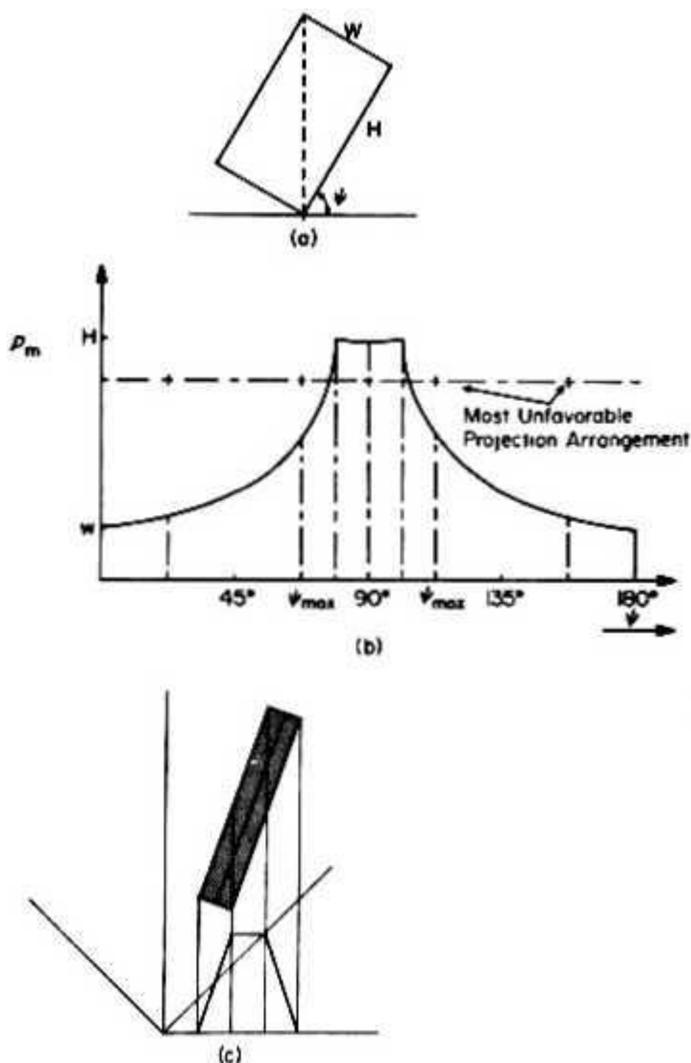


Figura 5.11 **(a) Definiția unghiului ^. (b) Variația valorii maxime în funcție de <. (c) Poziție în care niciuna dintre proiecții nu este aproape de dimensiunea maximă a obiectului.**

Ca aspect practic, este puțin probabil ca proiecțiile în sine să poată fi utilizate pentru recunoașterea caracterelor întregului alfabet, dar pot servi ca un preprocesor util. Multe caractere pot fi recunoscute strict din proiecțiile lor (de exemplu, L, T, I, H etc.), în timp ce pentru altele trebuie utilizate caracteristici suplimentare (de exemplu, R versus A).

5.5 NOTE BIBLIOGRAFICE

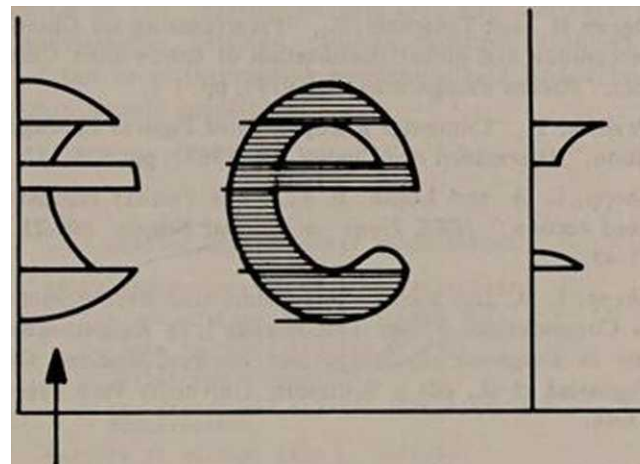
Abrevierea CAT se referă la utilizarea tehnicilor de reconstrucție pentru a determina și afișa o secțiune transversală a corpului uman din raze X laterale. Inițial, CAT a reprezentat *tomografia axială încrucișată,* dar acum este interpretată ca *tomografie asistată de computer.* Mașinile care fac reconstrucție din proiecții se numesc *scanere CAT* și constau dintr-un minicalculator, un hardware digital special pentru calcularea convoluției sau a transformărilor, afișaje grafice și alte periferice ale computerului, echipamente de raze X și piese mecanice elaborate pentru rotirea sursei de raze X și a receptorilor în jurul pacientului. Aceste scanere au revoluționat diagnosticul medical pentru că acum medicul poate privi o imagine, în timp ce în trecut informațiile necesare trebuiau obținute prin intervenții chirurgicale exploratorii sau prin injectare de lichide. O idee despre implicațiile medicale ale tomografiei poate fi văzută din articolele publicate în reviste de specialitate, cum ar fi *Journal of Computer Assisted Tomography* (New York: Raven Press). Pentru a vă asigura că toate fasciculele de raze X sunt paralele și pe același plan este necesar să existe o precizie mecanică ridicată în echipament, ceea ce se adaugă la costul total. Din acest motiv există interes pentru tehnicile de reconstrucție care utilizează fascicule neparalele.

Datorită importanței medicale mari a subiectului, cea mai mare parte a literaturii apare în legătură cu acea aplicație. Analiza Secțiunii 5.2 se bazează pe o lucrare a lui Shepp și Logan [5.SL]. Vezi [5.SS] pentru exemple suplimentare de probleme de reconstrucție bazate pe această metodă. Există o altă metodologie de reconstrucție bazată pe un alt principiu, faptul că evaluarea lui/(x,y) din proiecții poate fi exprimată ca soluție a unui sistem de ecuații liniare (5.HR). Volumul [5.HE] este o colecție de recenzii pe acest subiect, în timp ce lucrarea lui Budinger [5.BU] este o trecere în revistă concisă a reconstrucției .

[5.HA], în special [5.HH], conține o discuție despre anumite subiecte avansate în reconstrucția imaginii. Discuția metodologiilor din aceste referințe este suficient de generală pentru a fi de interes pentru cititorii a căror specialitate nu este microscopia electronică.

Proiecțiile oferă o modalitate ieftină de a efectua unele analize de formă. Utilizarea lor în recunoașterea caracterelor a fost descrisă pentru prima dată în (5.PAJ. Numai

Au fost utilizate proiecții verticale și orizontale, dar integrarea a fost efectuată separat după fiecare gol (vezi Figura 5.12). Cursurile au fost determinate printr-o combinație de detectare a pragului și a pantei, iar prezența lor în anumite locații a fost folosită pentru a genera caracteristici binare. Apoi au fost utilizate tehnici statistice standard pentru clasificarea finală. O altă aplicație a proiecției este descrisă în [5.NA] unde transformatele Fourier a patru proiecții au fost folosite pentru recunoașterea caracterelor chinezești. [5.OT] descrie utilizarea proiecțiilor împreună cu alte caracteristici.



Proiecția este calculată numai pe oart umbrit.

Figura 5.12 **Înmulțirea proiecțiilor integrale conectate [5.PA]. Figura prezintă proiecții de-a lungul direcției orizontale. Când se găsește un decalaj în timpul integrării, această valoare este dată primei proiecții (afișată în stânga mai sus), iar orice valori ulterioare sunt atribuite unei a doua proiecții (prezentată în dreapta sus).**

5.6 LITERATURA RELEVANTA

[5.BU] Budinger, TF „Computed Tomography: Three-Dimensional Imaging with Photons and Nuclear Magnetic Resonance", în *Biomedical Pattern Recognition and Image Processing.* **KS FuandT. Pavlidis, eds. Weinheim: Chemie Verlag, 1979, p. 179-212.**

(5.HA1 Hawkes, PW, cd. *Computer Processing of Electron Microscope Images.* **Berlin: Springer, 1980.**

|5.HE] Herman, GT, cd. *Imaginea k construcție din Projections.* **Berlin: Springer, 1980.**

[5.HH] Hoppe, W. și Hcgert, R. "Three-Dimensional Structure Determi nation by Electron Microscopy (Nonperiodic Specimens)," în [5.HA|, pp. 127-185.

(5.HR) Herman, GT și Rowland, SW „Trei metode de recunoaștere structing Objects from X-Rays: A Comparative Study", *CGIP,* **2 (1973) pp. 151-178.**

(5.NA) Nakimano, Y.: Nakato, K.; Uchikura, Y.; Nakajima, A., „Îmbunătățiți- ment of Chinese Character Recognition Using Projection Profiles," *Proc. First Intern. Joint Conf, on Pattern Recognition.* **Washington, DC (octombrie 1973) pp. 172-178.**

(5.OT) Ogawa H. și Taniguchi, K., „Preprocessing for Chinese character recunoașterea și clasificarea globală a caracterelor chinezești scrise de mână ", *Pattern Recognition* **11 (1979, pp. 1-7).**

|5.PA) Pavlidis, T., „Computer Recognition of Figures Through Decompo ţiunea," *Information and Control* **14 (1968), pp. 526-537.**

[5SL| Shepp, LA și Logan, BF, „The Fourier reconstruction of a head section." *IEEE Trans, on Nuclear Science.* **NS-21 (1974), pp. 21-43.**

[5SS) Shepp, LA și Stein, JA, „Artefacte de reconstrucție simulate în Computerized X-Ray Tomography", în *Reconstruction Tomogra phy in Diagnostic Radiology and Nuclear Medicine* **(MM Ter- Pogossian et al., eds. Baltimore: University Park Press, 1977, pp. 33-48).**

5.7 PROBLEME

5.1. Implementați algoritmul 5.1 folosind funcția dată de ecuație (5.(15) pentru *r(t)* (vezi și Anexa SA).

5.2. Să presupunem că știți că/(xj») are forma g(x)A(y), unde *g* și *h* sunt funcții necunoscute. Puteți folosi această proprietate pentru a simplifica algoritmul de reconstrucție?

5.3. Ni se dă o funcție J a două variabile sub formă tabelară: eșantioane *NW* . Vă puteți gândi la modalități de a utiliza proiecțiile pentru a reduce dimensiunea tabelului? Puteți lua în considerare două versiuni ale problemei: când o eroare este permisă în descrierea *lui* și când nu este. *(Sugestie:* ar trebui să faceți anumite ipoteze despre forma lui 5 și apoi să dezvoltați verificări pentru a verifica dacă funcția descrisă de un tabel dat satisface aceste ipoteze .)

5.4. Repetați analiza exemplului 5.1 pentru o regiune eliptică.

ANEXA SA: UN PROGRAM DE RECONSTRUCȚIE ELEMENTARĂ

Următoarea este o listă a programului FORTRAN și un exemplu de execuție a acestuia, implementând reconstrucția. A fost digitizat un afiș cu imagini și apoi proiecțiile au fost calculate prin însumare, urmată de reconstrucția cu și fără filtrare. Programul a fost scris de Michael Kass, ca parte a unei teme la Universitatea Princeton !

Consider că un astfel de proiect oferă studentului posibilitatea de a integra o mare parte din materialul din primele cinci capitole: digitizare, egalizare histogramă, filtrare etc. În plus, îi arată că unele dintre cele mai impresionante aplicații de procesare a imaginilor se bazează pe principii destul de simple.

```
laoe            0M>OU2.RECSN26,T-NO0,P-5OO,L«9999
C
C         IBAGE RICO «ST* OCT JO* F1 BICK
C         PROJECTIONS
C
C             I PROJECTIONS *82 C4LCDL4T1D POR AN N
C             PT N inxea. THEN THE ORIGINAL INICE is
C             RECONSTRUCTED er BACK PROJECTION WITH
C             FILTERED BED UNPZLTBBBD
C             PROJECTIONS.
C             ILL INAGES IBB PBlBTlt BITS HISTOGRAM
C             EQOALISATION.
C         WRITTEN NT BlZUABL KISS J/17/80
C
C
C         VABXA8LES
C
C     PXIlLtA.B)        ORIGINAL IH4GA
C     piinZcM)          PORTION OF IMG! TO PB USED
C     PNQJIA.8)         PROJECTION OP IR4CB ALONG ANGLE 4-PI/N BAjK“BOJEcilA'ol
C     HKjHl pepROj      TEE n?XLt£BBD PROJECTIONS BACK PROJECTION 0» THE
C     (l.h              FILTERED PROJECTIONS
C     AZ,AT             l.T COORDINATES OF 1MB BEGINNING POINT OP THE jtain? ss
C     BI,BT             $ina’M«?m5i°!i>K»«>o.
C     P1                RIDDLE OP THE XHAGB
C     P2                ^s'ws tnmtwnr »TM.M.O.
C     P3
C     CBTBT             USED TO GO PRO! CH4BACTE1 TO INTEGER REPRESENTATION
C     D CM
C     TIT
C
```

a fost rulat sub interpretul WATRV pe o mașină IBM 3300. Timpul său de execuție a fost de 162,5 secunde și avea nevoie de 17672 de octeți pentru codul său obiect și 155916 de octeți pentru matricele sale. Dimensiunile imaginii au fost 64X64.

^psw ^||^|!j5^v^^ •"""“' EÄS(;a«wi.i

BIAO IB PICTORB

£
c

PX1EL|I,J>>CNVE»«1

BEAD IB OflBPBIBTJBG CCD1S

DO 20 X«1,6
SHI^J 0 „^.™.«.«^

IBXTIAL11E

eu

39 i 1%;■ SIRAFLOAT 11-11 •3.H1593/PLOAT (Bl 1 COS^B?
j^-C^S {FLOAT (1-1) *5. H159 J/FLOAT (ND P1<HPU
(FLOAT (92)) .FLOAT |B2>)

l« fe 5 EL I CT A PCBTIOB OF TUB PICT«!

SS i: SV:

PIXEU (I, J) -PIXU(32«X. 1O«J1

F0UAT(/15b ORIGINAL 18101)

CALCULAȚI PROIECȚII

DO 67 J> 1,9
DO 67 K«1,B

FIRD I9DPOZBTS DE INTEGRAL

«55»»: <■> 2-(«-
T)«»2)

JMlHMi« [1]
BT-ilMG (P2-PJ) COBTI9ÜB

XBTEGBA11

SÜB-O.

II»«BU

FUTU
82 II 5:1;!
FPBOJ
DO"capac
I1-K-

FMdJ(l (J) -FPAÖJ (I ,J| -FBQJ1I.J-I1)/FLOAT (•• (R-5) ••2-1)

CALCULAȚI PROIECȚIILE SPATE

DO 106 1-1,B
DO 106 J-1.B

w3i!!M»aw:fiW"^^

91 IF(THITA 1.M.1.) șină* »•FLOAT (9) •THETA1

S c soa PE TOATE PICJICTIOBS

FBPBOJ I't J l'D
||J“c|spPL6At(XP9X)-fBETA1>*3.»MMJ/FLOAT(BD

A M
j

BB^tfiiHSKtWIIHHW^^

PBIBT BACK FBOJBCTBD IBAGES

107 OffW&H8 !k (WKJ? 1,1

108 poMyVlWttcoasTtaeTiOB lira OBFXLTMB» FROJBCTXOIS I

109 SK4lh(ta!H£JW., 1 sr

FoiSnWlK BSC0B5TB0C1I0B BITS FILTH ED PBOJ1CTIOBS)

110

120 ABD

92 suBiooTxaa FEI»T<PXIIL2.C, rotas,w n !H!H?

c(A,8) PENTRU XETiaSITT XF»»l 8. OFEBPRIBt 1(1,91 TNMOGH
aift'H A B0FF1B TC BOLD O>1 ^MF? ?■??RG^”” 1'taiwm«n»?,t*r-

e®' ii<i!^^^

PIUL(AB) CORTAIRS TM“S«'M*T.0F TRI COMIWOBDIBG PIXEL

BIST(A) COBfAIRS TIB BOBBER OF PIXELS BlTH XBimSITT A

R RsSRHJ'ISJMUBS'K«.«»»

HIRT
UV| (HA FIB)
eu?,» 11
SR“ 1 EA aiXXIOI PIXEL TALJE
„Ä1BDR FHIL FALDB

8
96

»^M

XBTKIBO PlX.Pl.Pi
COBPOTi BAI. MB BOB SCBLJBC

RIt-1003000 æ a
SVÏ riIaPXIBpj
EL ASIIQ [B. M*I-
AM*I0/RI

23

SCALB

Hxnjlj g-1*(HIBW (1 .J» -MB* 1) M 26./7LOAT

CALCULATE! H1STO08AH

Mif:^” 8
SS SS U
»!-■•■/» ^'^ 1 ' arsT <' 4,1,11 1 1 ' 4 » • 41

80

CALCULATE! BISTOGtAB BQOALXUUOB

?

PC^oS 8-1. 12®
LWt(l)"l BIBT-BIIT«HXST (!)

HIBT-HIBT-BAV!»!
■ ivTI^lîxGBTd) •LirT(S))/2

D0PUCBTB OVBBNXBTIBG MTMI

H 100
J

M>^W
DO 30 J»1,6
Cl(J,DC(J,Bl»(l)»1)

PBXBT ODT XBA68

«SM,»?»

30
C

HHJifcjta^^ „„ .„.„„
HS""

120

ü

e 6
u

m COBPLEI FU ICT 10« P0UI(IR|

CONCURSURI POLAB COOBDIBIT ESTE TC
BECTABCOLAB C IB TBB BOBB OF A COBBLII BOBBBB

1« g^hURsmnCTm» «.,

F UACT 101 BXVTBPiBAT.I

IWEB'WIWS#»

X?"(X1.LB.O) <1-1 IF (II. 6E.I) I1-B-» lijmf:« œi-i A^UT 11 »TU

IBTMPOLATIS XITO AB IirfEGIB HAT81I I.f IS
TBB POUT AT BHJCH TBI POrtlOI BBPIBSEBTID
BT TRI "4TUX I5 TO FE BVALOATBO. B XS TRI
DIHEBS1CB OP Til BATBXI.

C
C

sn Hdj-n» «ræ-Â BETOBB

BBC

```
      REAL FUNCTION INTRP2(MAT,X,Y,N)
      REAL*4 MAT(64,64)
      INTEGER X1,Y1,A,B,C,D
C
C        THIS FUNCTION IS THE SAME AS RINTRP
C        EXCEPT THAT THE MATRIX IS REAL.
C
      X1=X
      IF(X1.LE.0) X1=1
      IF(X1.GE.N) X1=N-1
      Y1=Y
      IF(Y1.LE.0) Y1=1
      IF(Y1.GE.N) Y1=N-1
      A=MAT(X1,Y1)
      B=MAT(X1+1,Y1)
      C=MAT(X1,Y1+1)
      D=MAT(X1+1,Y1+1)
      E1=A+(B-A)*(X-X1)
      E2=C+(D-C)*(X-X1)
      INTRP2=E1+(E2-E1)*(Y-Y1)
      RETURN
      END
```

SI
»TAT

«sttmnsstt
u?si:i2::KS!s

??8

Sie::
sssns
^555

Proiec
t de
lege

B-SI ih
110*8
IBOIH
10

sus^ns

wsum^:?;::^
i&i&W^^

0RIGIB4L IBÄGE

t:«fii
salut»

t-stni.T.nr.yjüSässss

;mw«:m

SUOII

IHSiiSISSHSSSSiSUSKSS
SSi&r.'.
ttuhnsssH^HSs^^

BK0IST8UCTI0B Bit« PROIECȚII FILTERID

:su:::;:nîi!î!!!:H!H;îS!:u:K!n!5x^

!t a at« a iooooo MOM »eooooo i a I* 11 » > » « » MM • - ■ « ■ ■ ■ fooo x x x • .-..
■ * a MI a a a a a a a a M a a M M eooooooo w ee « a a 8 a a a a a a
oo 0 o 1X OBMMWMMMOOOOCMMBBBBOMOOOOIIXX»

« • XI oooo ■ a ia I oocc ■ i«eeee9e«MMM
IIOOOMBBM OCOOCCMOOMtiiuatBie

ÛOÜÛOCCIIÏ8HllMHIHIIl»fiilMMll«IIIOQ,
DO mii I xxoooooocMveiM« aeavvesiBveMMM WHMsaao MOOOIII
iooo BBwaaiBaH«ftt*iioKVMb8Bve>iMis WMMBOOO

Hi

MMaaaaMMiMiMiiflMMooccooooM
MMM«M4M#a«i4i4icMMoooo iixaooo'

IOOO
IIII
•III
IIII

tlCOBSUUCt 108 MUM UIHUIBID P8OJEC1IOIS

OOOMBN8 8 8B640 8WHM8BOOOOOIIII»
IOOOOOB MMM8 88 »8880000 GOOII»»*»
II0000008V88N 8 »880001 111 1» XI III 11 008888 8
008888 8 8000 •••••• 1IIXII0CBB18C0001I»»»»»——
III1ICCB80C01I1»»»“»»“ -----------------. .
♦•IllIIOOO0011•••------------------------

--- -^*»•III»II^II•••••»-*•— ~ »•— —•••I iu» HI Uli
«♦♦♦♦♦•♦♦•“ ».-«* »»I II 1 1» XII1 UI IXXX IIXX»»»»»
.»«... »occi» xix .
Il 11X0001XIIX COM »BMIMOCGBIO

• IIIIOOBBÖIIOCCBaoeMBMMMBBMI XlooooBMOioocc MoeoeooMOOMMN

>xiiiiiioiii>iiviiccib)i>ee«!tiHifll9Vii(nil iiioaawMoMwtiaaataMeettMtttttoMo;
loooiiocBMeeMMMMMMMBOM»MOMOoea»BOi OOBOOOOCBN veMâvv ai IM ki BOMM IOMVOONVBOO;
* » 86111*16 88# 8001
8NM8MMVMB8WBBMtMMeMMMB»M6«M66MM»eW8M0II
aMoeeeeee8aMM«M6e>iiavwiBB106 »
■*MeMI4«BVBMM6MMlllMB16OO0IBiB6NWVMOIl
16 «068» 066««©«« 1118 B MI IM 110 611116 0 8 8 8 B OC 111011 Ku
M oooeoeeee I » 11 * 1 aMiMM 1111 ■ a io 8 aaaa« oo001 « 001 » 11 e
11 aiiM ia 11 a ia ie aaaaa ooooooooi 88880048 11 »1 II
Ml11188118*1111168 8»COOOCOOOOOO>
---------------------------------IMMOMMMBISMMOOOCCOOCOOO«
MMMCBBOOIIIXII MH IB «8
800001111 MMiieHNoiiixii«
«11688180000X11

«a a I lesa aioooooiI xi*»»
*110081888000111X1111
»11«»8M*800011 III 1111
■■■■ 11O» WBMUOOIXI»»»»»»»
titiaremiooxii»*«'«»'»'
Miieoisaioooixi»»»»»«—— tBlMQ BN
BBOOIII« •♦♦♦<— ■ iiiievBa »Doom» -
-- a-iiaito»
a-II - a-a-a-a-a-a -a MeMicooxxi»» ♦-—....
BlMBMOIVBBOOCII»»» -------- .....
■aaaazieo MMOOII»*«——....---•
1168 kB BOCIU •♦—. ,——— B1MM
MIO1IX»»—. . —— «IWïlBOOl»»—
.. --

Capitolul 6

STRUCTURI DE DATE

6.1 INTRODUCERE

Cerințele mari de memorie asociate cu stocarea datelor picturale sunt bine cunoscute de oricine a lucrat cu acestea. De exemplu, stocarea unui cadru obișnuit de televiziune necesită cel puțin 512x512 octeți, dacă folosim trei biți pentru două dintre culorile primare și doi pentru a treia. O fotografie de pașaport alb-negru necesită cel puțin o matrice de 64x64 cu șase biți pe element, cu mult peste dimensiunea unei înregistrări care conține orice alte informații se află într-un pașaport. (O pagină cu text dactilografiat la un singur spațiu necesită aproximativ 3000 de octeți.) Problemele de stocare, căutare, regăsire, transmitere etc. sunt deosebit de dificile ori de câte ori sunt întâlnite date picturale. Aceste dificultăți sunt contraintuitive, deoarece oamenilor le este adesea mai ușor să se ocupe de imagini decât de text. Ne este mult mai ușor să ne amintim chipul unei noi cunoștințe decât o pagină de text dactilografiat. Dificultatea de a potrivi o astfel de performanță umană pe computer poate fi apreciată subliniind că, cel puțin pentru unele persoane, rememorarea feței este mai bună atunci când aparține unui membru de sex opus și că textul este reținut mai bine dacă este o proză decât dacă este o listă de nume. Prin urmare, orice tehnici de compactare a datelor care depind doar de procesarea semnalului nu vor reduce volumul de date la o dimensiune compatibilă cu așteptările noastre intuitive. Utilizarea recunoașterii modelelor poate duce la o compactare mai mare (a se vedea Exemplul 1.1), dar o astfel de procesare necesită o cantitate suficientă de calcul, astfel încât cineva să fie încă obligat

pentru a face față (problema stocării și reprezentării imaginilor pe un computer.

Problema compactării datelor are diferite forme în comunicații și în calcul. În multe aplicații, o imagine este creată, transmisă, observată și apoi aruncată. Considerentul major în comunicații este reducerea *lățimii de bandă* necesară pentru transmisie sub constrângerea că procesarea trebuie să se facă în timp comparabil cu cel necesar pentru generarea și transmiterea imaginii. O situație diferită există în aplicațiile în care este necesară o bază de date picturale și imaginile trebuie stocate pentru perioade lungi de timp. Imaginile noi trebuie comparate cu cele vechi, sau seturi de imagini trebuie căutate pentru anumite caracteristici. Astfel, pe lângă volumul necesar pentru datele picturale, accesul poate fi o problemă.

Oamenii se pricep foarte bine la concentrarea pe părți ale unui afișaj bidimensional, dar computerele trebuie să procedeze într-un mod orb. Prin urmare, - algoritmii de traversare a imaginilor sunt foarte importanți. Pentru imaginile discrete, aceasta corespunde traversării unei grile discrete care poate fi interpretată ca un grafic ale cărui noduri sunt pixeli și ale cărui ramuri conectează noduri corespunzătoare pixelilor învecinați. Pe lângă această mapare simplă, se pot construi și alte grafice corespunzătoare unei imagini. La modă. deoarece toate structurile de date utilizate pentru imagini sunt grafice, este o idee bună să începeți cu o revizuire a algoritmilor de traversare a graficelor. (Cititorii care nu sunt deloc familiarizați cu teoria graficelor ar trebui să citească Anexa 6.A.) Există o restricție asupra unor astfel de algoritmi atunci când sunt aplicați graficelor care reprezintă imagini. Datorită dimensiunii lor, aceste grafice sunt rareori reprezentate în oricare dintre formatele obișnuite (de exemplu, matrice de adiacență, listă de noduri etc.). De exemplu, în cel mai simplu caz există doar coordonatele *xy* ale pixelilor și valorile lor de luminozitate sau culoare. Astfel, găsirea nodurilor adiacente unuia dat este o problemă reală. Dacă este necesar să traversăm doar acele părți ale unei imagini care formează o regiune uniformă, atunci trebuie să decideți dacă pixelii vecini aparțin aceleiași regiuni cu pixelul curent. Din aceste motive vom lăsa termenul • adiacent” în mod deliberat vag, cel puțin în acest capitol Presupunem doar că există o procedură *ADJACENT(p ,n ,N)* care pentru nodul (sau pixelul) p returnează numărul *n* de noduri adiacente care nu au fost vizitate anterior în timpul parcurgerii, iar tabloul *N* conținând coordonatele lor (coordonatele xy).

6.2 ALGORITMI TRAVERSAȚI GRAFICI

Parcurgerea unui graf conectat este un proces simplu dacă se începe dintr-un punct și apoi se trece la cele adiacente acestuia. în timp ce se marchează orice loc care a fost deja vizitat. Dacă există mai multe puncte adiacente, atunci unul dintre ele este ales ca următorul loc de vizitat și

celelalte sunt puse deoparte, pentru a putea fi folosite ulterior pentru a relua traversarea. O structură convenabilă pentru salvarea acestor puncte este o *stivă.* Aceasta este o matrice în care sunt adăugate elemente în timpul căutării și. când

t De exemplu, vezi pp. 44-49 din [6.AHU1. sau pp. 234-304 din |6.KN],
t De exemplu, vezi pp. 176-179 din (6.AHU], sau pp. 51-57 din [3.PA|.

algoritmul o cere. elementele sunt eliminate. Regula esențială este că ultimul element adăugat este eliminat primul. Deoarece vom întâlni aceste operații de multe ori în continuare, le dăm nume speciale: *PUSH(p,S). zd<}* pixel p pentru a stivui S'; și *p = POP(S),* eliminați un punct din *S* și numiți- *I p.* Algoritmul 6.1 prezintă două programe simple care implementează aceste funcții. Valoarea „I este folosită ca indicator al unei stive goale. Dacă este o adresă legitimă într-o anumită implementare, atunci trebuie returnată o altă adresă imposibilă pentru a indica o adresă goală.

Rețineți că *S* nu trebuie să fie o matrice unidimensională, mai ales în contextul procesării imaginii. Acolo, graficul corespunde planului imaginii, cu pixeli pentru nodurile sale, iar nodurile adiacente corespunzând pixelilor adiacente. Pentru a descrie un punct trebuie să stocăm atât coordonatele lui x cât și *y ;* astfel. *S* va fi o matrice de perechi.t

Algoritmul 6.1 Proceduri de manipulare a stivei

Notație: i este indicele ultimului element al tabloului *S.*

Procedura *PUSH(p J)*
L Creșteți r cu I și setați S(f) — p.
2. Sfârşit.

Procedura *POP(S)*
J. **Dacă** este mai mare decât 0. **atunci** se reduce *i* cu 1 și **se întoarce** S(i+1).
3. **else return** —I pentru a indica o aspirație goală.
4. **Sfârşit.**

În timpul traversării este necesar să se utilizeze două tipuri de marcaje. Presupunem, fără pierderi de generalitate, că inițial toate nodurile (pixelii) sunt marcate cu unul. Când un nod (pixel) este plasat pe stivă este marcat cu doi, iar când este străbătut este marcat cu zero. †Un algoritm de parcurgere a grafului conectat de bază (regiune) care utilizează aceste funcții este listat ca Algoritmul 6.2. De asemenea, ar trebui să subliniem că wc poate omite

t Aceasta presupune că imaginea este stocată ca o matrice bidimensională. Implementarea unor astfel de matrice este incomodă în multe limbaje de calculator, deoarece datele sunt stocate intern într-un mod unidimensional, h este cel mai bine să stocați imaginile în mod explicit ca matrice unidimensionale și apoi să folosiți pointeri pentru a aborda pixelii individuali. Într-un astfel de caz, stiva va fi o matrice de pointeri (sec Problema 6.1).

marcarea pixelilor atunci când sunt plasați în stivă. O astfel de omisiune nu va afecta corectitudinea traversării (vezi problema 6.2), dar poate provoca o creștere mare a dimensiunii stivei. Se pot construi exemple în care pentru un grafic cu *N* noduri dimensiunea stivei se apropie de *N .* Cu toate acestea, în multe aplicații, forma graficului este de așa natură încât dimensiunea sutului nu este o problemă (vezi Secțiunea 8.4.4).

Algoritmul 6.2 Algoritmul de traversare a graficelor conectate de bază

Notație: **Procedura** *ADJACENT(p ,n ,N)* **primește locația pixelului** *p* **și returnează numărul vecinilor săi** *n* **și locațiile acestora, stocate în tabloul** *N .*

0. Găsiți un nod (pixel) *p* al graficului (regiunii) de parcurs.
1. Puneți *p pe S* suge .
2. **În timp ce** *S* nu este gol, **faceți** pașii 3-9.
 ÎNCEPE.
3. *p~POP(S).*
4. **Repetați** pașii 5-9.
 ÎNCEPE.
5. Marcați *p* și **apelați procedura** *ADJACENT(p,n,N).*
6. **Dacă** *n* este egal cu 0, **atunci** ieșiți din buclă.
7. Setp=/V(l).
8. **Dacă** *n* este mai mare decât I, **atunci faceți** pasul 9.
 ÎNCEPE.
9. Pentru i = 2 la n *PUSH(N(i)S).*
 Sfârşit.
 Sfârşit.
 Sfârşit.
10. **Sfârșitul algoritmului.**

În timp ce suck este o structură dau convenabilă pentru traversarea regiunii, există alte probleme în care se dorește eliminarea elementelor dintr-o matrice în ordinea în care au fost plasate acolo. Această structură dau se numește *coadă.* O funcție *ADD(p ,Q),* adaugă pixelul *p* la coada 0, este similară cu funcția *PUSH* pentru stivă. Funcția *REMOVE (Q)* care elimină un punct dintr-o coadă necesită mai multă atenție. Algoritmul 6.3 enumeră aceste funcții. Rețineți că sunt utilizați doi indici, indicând începutul și sfârșitul cozii. Această implementare este simplă, dar din moment ce irosește spațiu, trebuie să realocați periodic spațiul de stocare.

Pentru mai multe detalii despre implementarea și gestionarea stivelor și cozilor, cititorul este referit la oricare dintre numeroasele cârlige ale algoritmilor și structurilor de date (vezi și Notele bibliografice), f Dacă am
înlocuiți stiva cu o coadă în algoritmul 6.2, graficul este încă parcurs corect, dar nodurile sunt vizitate într-o ordine diferită decât înainte. Parcurgerea corectă este posibilă și fără niciuna dintre aceste construcții de stocare prin utilizarea backtracking atunci când nu mai este posibilă avansarea. Toți acești algoritmi de traversare a graficelor sunt discutați în detaliu în multe texte.t

Algoritmul 6.3 Proceduri de manipulare a cozii

Notație: tendința este indicele ultimului clement al tabloului *Q.*
istan este indexul primului element al lui *Q.* Inițial ambele sunt egale cu prima adresă a spațiului matrice.

Procedura *ADD(p,Q)*

1. *Tendinta* de creștere cu 1 și se stabilește *Q(iend) = p.*
2. Sfârşit.

Procedura *EXTRAGERE (Q)*

1. **dacă** *istart* nu este mai mare decât *tend,* **atunci** incrementează *istart* cu 1 și **returnează** *Q (istart—]).*
2. else return —I pentru a indica o coadă goală.
3. **Sfârşit.**

6.3 PAGINARE

Cea mai simplă modalitate de a prezenta imagini din clasa 1 este prin matrice întregi. Dacă dimensiunea matricei o depășește pe cea a memoriei rapide disponibile, poate fi necesar să o subdivizați în *pagini.* Acesta pare a fi un proces banal, cu excepția faptului că se poate întâmpla ca linia care desparte două pagini să fie aproape de o parte interesantă a imaginii. Pentru a evita schimburile repetate de pagini, sunt adesea folosite pagini suprapuse. Figura 6.1 prezintă un aranjament posibil. O astfel de suprapunere este de dorit nu numai pentru a minimiza schimburile, ci și pentru a detecta caracteristicile care nu pot fi văzute decât dacă există o zonă suficient de mare a imaginii în jurul lor.

Este de dorit să stocați astfel de pagini pe un disc sau bandă în așa fel încât proximitatea geometrică a paginilor să corespundă proximității în locația lor de stocare. Se poate demonstra că acest lucru este imposibil în general (vezi problema 6.5). Apropierea geometrică poate fi păstrată numai cu privire la o anumită pagină. Figura 6.2 prezintă un exemplu. Pagina critică este 5 și toate paginile care au o chenar cu ea sunt stocate mai aproape de ea decât paginile care nu o au. Astfel de strategii pot fi utile în alocarea stocării intermediare. O imagine este stocată pe bandă, o pagină este adusă în memoria rapidă, iar paginile din apropiere sunt stocate pe un disc într-un mod similar cu cel sugerat de Figura 6.2. Ar trebui să știți că mulți

sistemele de operare au propriile lor strategii de alocare a spațiului și că nu vor. în general, păstrați adiacența geometrică. În astfel de cazuri, este necesar să ocoliți sistemul de operare și să tratați discurile ca dispozitive de date brute.

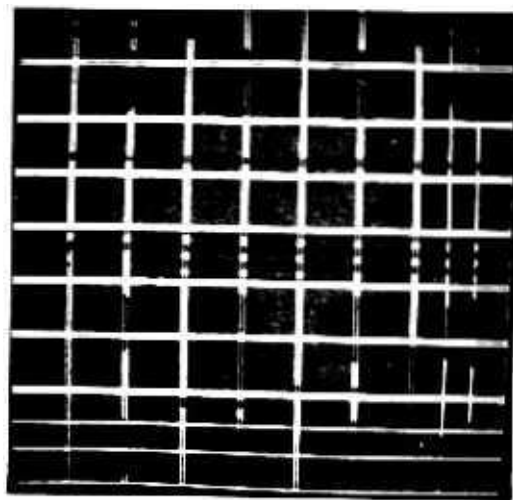
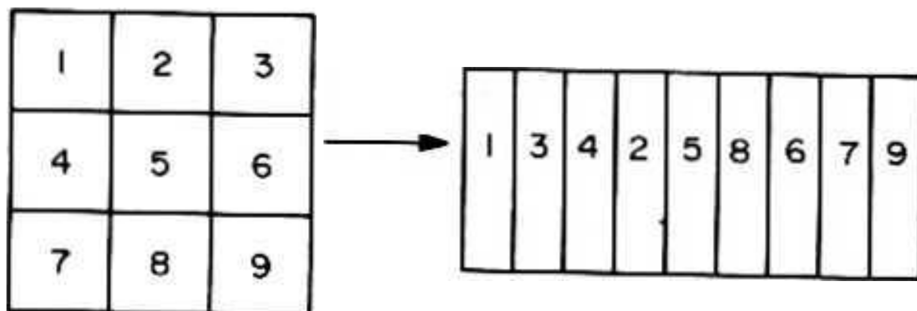


Figure 6.1 **Paging arrangement: overlay of pictures on an image of a primed wiring board**



Fi 1U r« « 2 Dispoziție de depozitare pentru păstrarea adiacenței geometrice în raport cu pagina 5

6.4 PIRAMIDE SAU CADRUL COACI

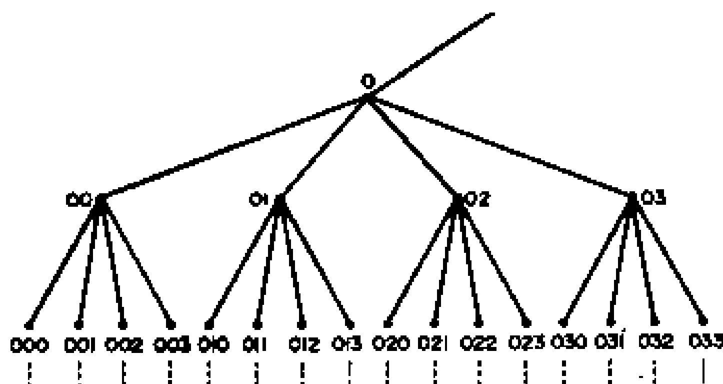
Piramidele sau arborii quad sunt o structură de date populară atât în graficele, cât și în procesarea imaginilor. Această tehnică este utilizată cel mai bine atunci când imaginea este o matrice pătrată 4 a cărei dimensiune este o putere de 2, să zicem 2'*. Atunci /I poate fi subdivizat în patru matrici pătrate ^ 0 ^i^b^j. ale căror dimensiuni sunt

jumătate din *A.* Acest proces poate fi repetat recursiv *de n* ori, până când este atins nivelul unic de pixel. Nivelurile pot fi numerotate . începând cu zero pentru întreaga imagine, până la *n* pentru pixelii unici. Un anumit pătrat poate fi etichetat cu unul dintre simbolurile 0. 1, 2. sau 3, concatenat cu eticheta pătratului său predecesor. În acest fel, pixelii unici vor avea etichete lungi de *n* caractere. Putem exprima acest aranjament ca un *arbore,* ale cărui noduri corespund pătratelor. Nodurile sunt conectate dacă unul dintre pătratele corespunzătoare îl conține imediat pe celălalt.

Rădăcina copacului corespunde întregii imagini, frunzele pixelilor unici și toate celelalte noduri au gradul inferior 4. Un astfel de copac este de obicei numit *arbore cuartic* sau *arbore quad.* Figura 6.3 prezintă un astfel de arbore și Figura 6.4 arată notația de adresare pentru o imagine 8X8. Deoarece nivelul *k'** conține 4* pătrate, arborele are un total de

N~Z 4* - 4"

noduri. Prin urmare, există cu aproximativ 33% mai multe noduri decât pixeli. Se pare că pentru multe aplicații arborele poate fi stocat folosind doar locații de 4". Discutăm în continuare despre crearea unui astfel de arbore quad. x



Figart 6.3 **Parte dintr-un arbore quad**

000	**001**	**010**	**011**	**100**	**101**	**110**	**111**
002	**003**	**012**	**013**	**102**	**103**	**112**	**113**
020	**021**	**030**	**031**	**120**	**121**	**130**	**131**
022	**023**	**032**	**033**	**122**	**123**	**132**	**133**
200	**201**	**210**	**211**	**300**	**301**	**310**	**311**
202	**203**	**212**	**213**	**302**	**303**	**312**	**313**
220	**221**	**230**	**231**	**320**	**321**	**330**	**331**
222	**223**	**232**	**233**	**322**	**323**	**332**	**333**

Figura 6.4 **Exemplu care prezintă sistemul de adresare a pixelilor unei imagini 8X8 conform**

unui arbore quad: prima cifră identifică nodul de primul nivel, a doua cifră nodul de al doilea nivel și așa mai departe.

6.4.1 Crearea unui arbore cvadru

Presupunem că începem cu o imagine care poate fi accesată rând cu rând. Acesta este cazul când este stocat pe o bandă sau disc, sau când este digitizat de un dispozitiv de scanare a tamburului sau unul bazat pe o cameră de televiziune (vezi Secțiunea 1.3). Pentru a forma arborele, citim în două astfel de rânduri și apoi procedăm la examinarea cvadruplelor de pixeli în ordinea prezentată în Figura 6.5a. Pentru fiecare grup de patru pixeli cu indici 2*. 2*+l, 2"+2A 2"+2*+l (*=0,1,2 ■ ■ • . 2"" ' -l), fie / 0 . A- *f** și /, să fie nivelurile de gri corespunzătoare. Putem crea patru niveluri noi

Deci 2// (6.1a)

gj^f)-gQ j-1,2, 3 (6.1b)

Rețineți că nu există nicio pierdere de informații în trecerea de la //s la g/s deoarece putem recupera //s din g/s prin formule

/>"^o J-1,2, 3. (6.2a)

3
/o - *o“S& • (6.2b)
/-l

Singura problemă este că poate necesita mai mulți biți pentru a reprezenta g/s decât /y. Vom reveni la această întrebare mai târziu.

0	1	4	5	10	11	14	15	20	21	
2	3	6	7	12	13	16	17	22	23	
100	101	104	105	110	111	114	115	120	121	
102	103	106	107	112	113	116	117	122	123	* •

(o)

9 USD	£4				24 USD	

(^I.g2.g3>(g5.^7) ' ' • <£?$>£ ?6.£7?)

(b)

merge	4 dolari	10 GBP	14 GBP	20 GBP	24 GBP	£.«o	
Sioo		£11Q_	£ IM	£129- ■	124 GBP	130 GBP	

(C)

Figura 6.5 **(a) Ordinea în care pixelii sunt citiți dintr-o matrice pentru a forma un arbore quad. Etichetele arc în octal presupunând un 64 $_{10}$ X64| $_{0}$ imagine, (b) Rândul de sus arată un singur rând produs din două rânduri din imaginea originală, iar rândul de jos arată matricea diferențelor, (c) Elementele primelor două rânduri ale noii imagini.**

Pentru a forma următorul nivel al arborelui, creăm o nouă imagine cu un rând ale cărui elemente sunt go's- În același timp, plasăm într-o matrice separată *D* valorile diferențelor g $_b$ g $_b$ £3- Astfel. wc ajunge la forma lhc din figura 6.5b. Când se citesc următoarele două rânduri, indicele primului clement va crește cu 2' $^{1 * 1}$, unde 2" este lungimea unui rând, astfel încât va rezulta aranjamentul din Figura 6.5c. La sfârșit vom fi creat o nouă imagine de 2"""X2" $^{-1}$ plus o matrice *D* cu diferențe de 3-2" în continuare. Rețineți că spațiul total este încă diferențat.

27»"24.3.?2^—2 -a 2 ?

în ceea ce privește numărul de pixeli. Procedura se poate repeta pana ajungem la o imagine formata dintr-un singur pixel. Apoi aproape toate informațiile vor fi conținute în tabloul *D.* Procesul de creare poate fi descris de algoritmul 6.4. care folosește procedura enumerată ca Algo ritm 6.4a.

Algoritmul 6.4a Crearea unui Le»el al unui arbore cvadru

Procedura *TLEVEL(n ,1J ,D)*

Notație: **intrarea este imaginea *I* cu dimensiunea 2"X2". Ieșirea este imaginea *J* cu dimensiunea 2'" ^{I}X2'" ^{1} și matricea *D* care conține diferențele.**

1. **Pentru** *j* de la 0 la 2" cu 2 **faceți:**

ÎNCEPE.

2. Plasați rândul /* al imaginii *I* în tabloul *p* și rândul Q'+I) ^{rt} în tabloul *p* $_{V}$

3. Pentru i —0 la 2" cu 2 faceți:

ÎNCEPE.

4. Set/ $_0$ egal cu Pi(O.

/i topji+l), *fi* topjp), *fy* topiO+l).

5. Calculați go, g $_H$ *g* $_{it}$ fj prin ecuațiile (6.1).

6. Adăugați g $_0$ $^{10\ thc}$ imagine /.

7. Adăugați *g* $_($, *g* $_2$. *g* $_3$ la matricea *D.*

Sfârşit.

Sfârşit.

8. **Sfârșitul procedurii.**

Algoritmul 6.4 Crearea unui arbore cvadru

0. **Citiți** imaginea *I„* de dimensiunea 2"X2*.

1. Nivel stabilit = *n.*

2. **În timp ce** *(nivelul* este mai mare decât 0) **faceți:**

ÎNCEPE.

3. Apelați *TdJtVEL (nivel JMJM-I^*

4. Reduceți *nivelul* cu 1.

Sfârşit.

5. **Scrie/ $_0$,** *D* $_h$ *D* $_2$, - - $_t$ *D*.*

6. **Sfârșitul algoritmului.**

6.4.2 Reconstruirea unei imagini dintr-un arbore cvadru

Dacă o imagine este stocată ca matrice, transmiterea și afișarea acesteia se desfășoară pe rânduri sau linii de scanare. Fiecare parte este afișată în detaliu și dacă *T* $_e$ indică timpul necesar pentru afișarea completă, în 7^/2 putem vedea doar jumătatea superioară a imaginii. Acesta este un mod nesatisfăcător de a naviga prin imagini, mai ales dacă *T* $_e$ este mai mult de una sau două

secunde lungi. Ar fi mai bine să afișați în *T* $_{e/1}$ întreaga imagine la o rezoluție grosieră, astfel încât privitorul să poată decide rapid dacă există ceva interesant în ea și dacă nu trebuie să întrerupeți afișarea.

Algoritmul 6.5a Procedura *FILL(i ^^^^ y $_t$)*

1. *>"(^y,x,/J. {definiți fereastra pe ecran)

2. *spate(i).* {setează culoarea de fundal)

3. *șterge,* {terge zona ferestrei la intensitatea fundalului)

4. Sfârșitul procedurii.

Algoritmul 6.5 Afișarea imaginii cu informații brute mai întâi.

Notație: intrarea este matricea *D* ale cărei elemente sunt diferențe față de media locală, cu excepția primului element, care este egal cu media pentru întreaga imagine.

1. Setați / la primul element al matricei *D* și executați *FI LL(i,Q,0,2”,2”)*

2. Pentru *i* de la 0 la n—1 **faceți:**

ÎNCEPE.

3. Setați / - 4' și *u* - 2'^.

4. Pentru / — 0 la /— eu fac:

ÎNCEPE.

5. Citiți următoarele trei elemente ale lui D: g L g 2> g 3 .

6. Împărțiți *j* la 2 / și fie *q* câtul și *r* rest. Lely = *2qu* și *x — 2ru.*

7. Citiți *g^* din imaginea din locația (xj*) și calculați *fth ft* fi* fi* din ecuațiile (6.2).

8. Apelați *FILL(fo,x* j *jc+u* j'+w).
Apelați *FILLfJ It x+uj ,x+2u j+u).*
Apelați *HLL(f 2 ,x,y+u,x+u,y+2u).*
Apelați *F/LL(f* Jr r+uy+u,x+2M 1 y+2u).

Sfârşit.

Sfârşit.

9. Sfârșitul algoritmului.

Algoritmul 6.5 descrie procesul. Utilizează o procedură *FILL* pentru afișarea reală a imaginii și presupune că imaginea poate fi citită și înapoi de pe dispozitiv. Această procedură poate fi implementată cu ușurință cu comenzile grafice primitive descrise în Secțiunea 1.7 (Tabelul

f Nu este nimic special la *T,/l.* **Am fi putut lua în considerare orice altă fracțiune mică .**

1.2) . Dacă un afișaj grafic este conectat la un computer gazdă, aceasta este o metodă eficientă numai atunci când dimensiunea ferestrei este mare. Pentru dimensiuni mai mici, cel mai bine este să tamponați rezultatul creând o matrice în memoria principală și apoi scrieți cu o comandă *wpic* (Tabelul 1.2). Atunci când trebuie să comutați de la o tehnică la alta, dimensiunea critică a ferestrei este determinată prin compararea: (a) încărcarea magistralei mașinii gazdă la transmiterea instrucțiunilor pe afișaj, (b) viteza relativă pentru executarea unei comenzi *wpic* față de o secvență a celor trei comenzi simple de *FILL* de către controlerul de afișare și (c) overhead pentru crearea și manipularea matricei buffer de către computerul gazdă .

6.4.3 Compactarea imaginii cu un arbore cvadru

Reprezentarea descrisă în cele două secțiuni anterioare poate fi îmbunătățită în mai multe moduri. Prima noastră preocupare este să eliminăm nevoia de stocare crescută (sau posibilitatea unor erori de rotunjire) pentru *gf -urile* *definite* de ecuațiile (6.1). Ceea ce avem nevoie este o tehnică pentru o mapare unu-la-unu a patru numere pe alte patru.

Propoziţia 6.1: Fie a h a 2 • ■ • a„ n numere, fiecare constând din *Q* biţi. Se presupune că numărul n este o putere de doi. Atunci media lor plus diferenţele n —I ale O 2 , • • • *D„* din acea medie pot fi stocate în n0+log 2 n biţi.

Dovada: Fie *L* suma celor *n* numere. Fie *q* câtul său cu *n* și *r* restul său. Fiecare dintre cele *n* numere *q, a^q (i* = 2,...n) necesită cel mult *Q* biți. Restul *r* necesită cel mult logy: biți. □

Propunerea 6.1 arată că pentru arborele quad trebuie să adăugăm doi biți la fiecare nivel, iar acest lucru se poate acumula cu siguranță. Cu toate acestea, există un compromis între biții pentru *q* și *r* și *a t—q.* Dacă numerele *a,* sunt aproape egale între ele, atunci *r* și *a t—q* vor necesita foarte puțini biți, dar *q* poate necesita până la *Q.* Dacă există diferențe mari între afs. atunci *q* va fi mic și va necesita mai puțini biți. Această caracteristică poate fi exploatată pentru a produce reprezentări care nu necesită spațiu suplimentar și este discutată în secțiunea următoare. Aici arătăm cum rezultatul Propoziției 6.1 poate fi folosit pentru creșterea eficienței. În loc de a transmite diferențele de la medie, se pot trimite doar diferențele de la *q* (fiecare necesitând cel mult *Q* biți), plus ultimul *q.* În plus, putem transmite restul *r,* fiecare necesitând doar log 2 n biți (doi biți pentru arbori quad). Se poate găsi cu ușurință că numărul de resturi este

$$\frac{4''\text{-I}}{3}$$

prin urmare, creșterea volumului nu va fi de 33 la sută, ci de (2/0) ori de 33 la sută. Reconstructia se poate realiza printr-o simpla modificare a ecuatiilor date.

Este posibil să se realizeze economii considerabile dacă imaginea are zone mari uniforme. Într-adevăr, să presupunem că cele trei diferențe pentru un bloc sunt zero. Apoi, în loc să plasați trei zerouri în tabloul *D* folosit de algoritmul 6.4a, se poate introduce un singur simbol special. (Acesta ar putea fi valoarea maximă posibilă a luminozității, deoarece această valoare nu va apărea ca diferență față de medie). După ce scanarea întregii imagini a fost finalizată, repetările acestui simbol pot fi șterse din toate nivelurile ulterioare ale arborelui. De exemplu, o imagine cu o valoare de luminozitate va fi codificată ca acea valoare plus simbolul special (vezi Problema 6.6).

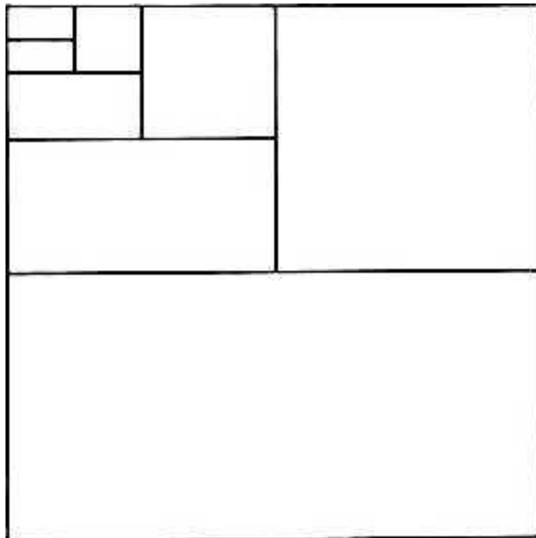


Figura 6.6 **Secvența subdiviziunilor imaginii pentru arborele binar**

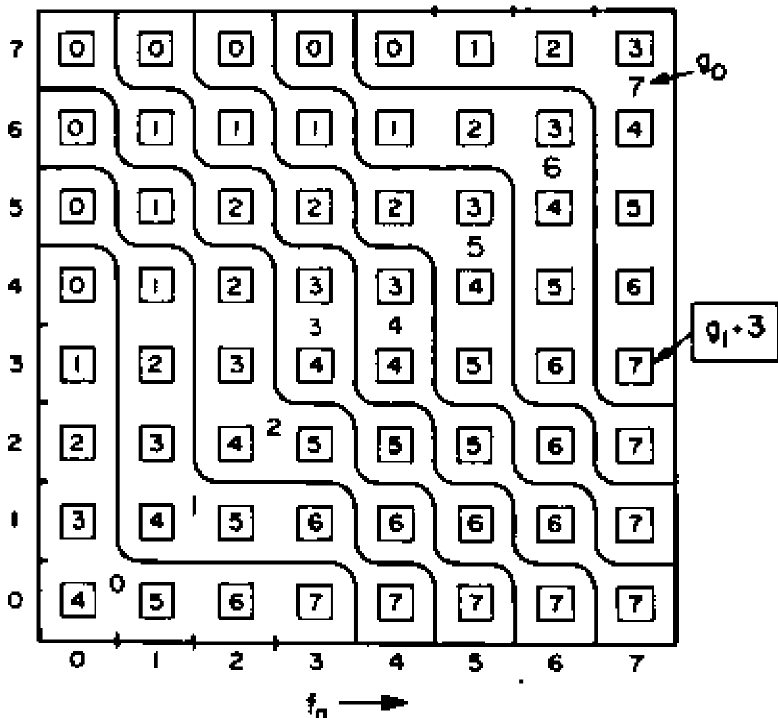
6.5 ARBORI IMAGINII BINARI

Arborii binari sunt o variație utilă a arborilor quad și au anumite avantaje care sunt descrise mai jos. Deși au fost propuși doar recent (vezi Notele bibliografice) și nu au fost studiate la fel de mult ca arborii quad, se poate considera că, pe termen lung, sunt o structură de date superioară. Figura 6.6 prezintă aranjamentul blocului

pentru arborele binar. Schimbarea formei blocurilor de la nivel la nivel nu este o problemă serioasă: se poate determina prin testarea dacă indicele de nivel este par sau impar.

Un avantaj clar al arborilor binari este modificarea mai mică a rezoluției de la nivel la nivel. În timp ce pentru arbori quad fiecare transmisie dublează rezoluția, pentru arbori binari fiecare transmisie doar o dublează. Deoarece volumul total de date este același în ambele cazuri, se formează mai devreme o imagine mai precisă.

Un alt avantaj major este utilizarea compromisului dintre medie și diferență, astfel încât să nu se transmită mai mulți biți decât în imaginea originală. În special, g_0 și g_t nu sunt definite prin ecuațiile (6.1) ci dintr-o aproximare discretă așa cum se arată în Figura 6.7.



Figare 6.7 **Relația care determină g_0 și $g_($ din / $_0$ și / $_($. Numerele din casete sunt valorile lui g_b cu 3 adăugat la ele, deci sunt încă în intervalul 0-7. Ele sunt aproximativ egale cu /o-/i + 3. Pătratul este subdivizat în regiuni în care numerele neboxate dau valoarea de g + $_0$ ca medie de aproximare a /J/2.**

Figura 6.8 (Planca 24) prezintă o secvență de imagini produse prin această metodă. Imaginea din stânga sus are o singură intensitate, media tuturor pixelilor. Prima imagine a celui de-al treilea rând are 4096 de blocuri distincte și este transmisă într-o optime din varul ultimului (a patra), care are 32.768 pixeli. Majoritatea informațiilor sunt deja disponibile în primul și, prin urmare, se poate decide rapid dacă să continue transmiterea.

6.6 ALGORITMI DE DIVIZARE ȘI-UNIUNE

Compactarea și accesul selectiv la imagini sunt doar două dintre cele mai comune utilizări ale arborilor quad (și binari). O altă aplicație majoră a acestei structuri de date a fost în segmentarea imaginilor. Secțiunea 4.4 a descris utilizarea testelor de creștere și uniformitate a regiunii pentru segmentarea imaginii . Algoritmii de împărțire și îmbinare sunt o extensie a acestei abordări. Se începe cu nodurile (pătratele) la un nivel intermediar al unui arbore quad. Dacă se constată că un pătrat este neuniform, atunci este înlocuit cu cele patru subpătrate ale sale (divizat). În schimb, dacă se găsesc patru pătrate care formează un pătrat uniform, acestea sunt înlocuite cu acesta (merge). Acest proces poate continua recursiv până când nu mai sunt posibile divizări sau îmbinări. Algoritmul 6.6 implementează această idee și o vom discuta după stabilirea

unei metode de descriere a nodurilor arborelui quad.

Rețineți că aici arborele nu este folosit pentru a transforma o imagine, ci pentru a ghida traversarea. Astfel, la nivelul *q* cu pătrate care conțin 2'X2' pixeli, calculăm criteriul de uniformitate pe toți pixelii fără a folosi valoarea medie a pătratului. Rezultatul este din nou o imagine cu rezoluție variabilă, dar acum avem o singură copie, mai degrabă decât o secvență de copii ca înainte. Figura 6.9 (Plansa 25) prezintă un exemplu.

Se pot construi algoritmi care încep de la un singur pixel și fac doar operații de îmbinare. Se pot construi, de asemenea, algoritmi care încep de la imaginea completă și fac doar operații de divizare. Primele au dezavantajul că este imposibil să se testeze multe criterii de uniformitate pe pixeli unici. Acestea din urmă au dezavantajul calculării excesive deoarece operația de împărțire este mai costisitoare decât operația de îmbinare (vezi problema 6.8). Începând de la un nivel intermediar ne permite să evaluăm predicate de uniformitate complicate fără a face un număr excesiv de împărțiri.

Implementarea structurii de date trebuie să se ocupe de problema stocării pătratelor de dimensiune variabilă. Este necesar să folosiți un antet pentru fiecare dintre ele, indicând dimensiunea și locația în termeni de coordonate *xy* pe plan. Dacă dimensiunea imaginii este 2'X2'', atunci *n* biți sunt suficienți pentru fiecare dintre aceste cantități. Astfel, 3n biți pe pătrat pot fi utilizați pentru a descrie coordonatele colțului său din stânga sus.

plus lungimea laturii sale. Se poate folosi și sistemul de indexare descris în Secțiunea 6.4. În acest caz, antetul va consta dintr-o etichetă lungă de cel mult *n* simboluri. Deoarece fiecare simbol este unul dintre numerele 0, l, 2 sau 3. poate fi descris prin doi biți. Totuși, furnizarea de informații despre lungimea etichetei necesită log^ biți per intrare. Etichetele cu lungime fixă pot fi obținute prin completarea cu un alt simbol, de exemplu, 4, care va crește spațiul necesar la trei biți per simbol. Prin urmare, în funcție de sistemul utilizat, antetul necesită 3« sau (2n, + log $_2$ n) biți, unde n_t indică lungimea unei anumite etichete. Deși anteturile cu etichete de lungime variabilă necesită mai puțin spațiu, creșterea complexității implementării probabil nu merită economiile în memorie.

Dacă numărul de biți necesar pentru a descrie caracteristicile unui pătrat este *m* și există /V pătrate, atunci spațiul total necesar va fi A(3w+m). Valoarea lui *m* poate fi destul de mare atunci când regiunile sunt caracterizate de multe caracteristici. Pe de altă parte, dacă singura caracteristică utilizată este luminozitatea medie, *m* este aceeași cu cantitatea necesară pentru descrierea pixelilor individuali. Se poate calcula un raport de compactare ca

$$\frac{4'' m}{N\{3n + m)}$$

Opt este o valoare tipică atât pentru *n,* cât și *pentru m.* astfel încât imaginea cu rezoluție variabilă necesită mai puțin spațiu dacă numărul de pătrate este mai mic de un sfert din numărul de pixeli. Acesta este adesea cazul, astfel încât această implementare nu impune cerințe suplimentare de memorie.

Rezultatul unui algoritm de împărțire și îmbinare care utilizează un arbore quad

necesită procesare ulterioară: fuziunea regiunilor care nu sunt frați în arbore. Pentru aceasta avem nevoie de o altă structură de date, care va fi descrisă în Secțiunea 6.7.

Algoritmul 6.6 enumeră procedura de parcurgere de bază a unei imagini de 2"X2" începând de la nivelul *q,* care conține 4' pătrate de dimensiunea 2" -? X2' '–' fiecare. (Astfel, întreaga imagine corespunde nivelului zero, iar pixelii unici corespund nivelului n *)* . arbore quad, aranjat în așa fel încât patru noduri succesive să aibă un părinte comun în arbore. Indicele utilizat pentru parcurgerea listei este exprimat în baza 4, astfel încât coordonatele xy ale colțului din stânga sus al pătratului să poată fi găsite imediat (vezi Figura 6.4 Într-adevăr, dacă *J* o , J b - (6.3), prezentat mai jos.

Algoritmul 6.6 Împărțire și îmbinare folosind un arbore cvadru

Notație: **Lista legată** *L* **conține coordonatele colțului din stânga sus și dimensiunea pătratelor plus indicatorii către elementele anterioare și următoare . Indicele /* este exprimat în baza 4. Intrarea este un set de pătrate corespunzătoare unui nivel** *q* **al arborelui quad. Ieșirea este un alt set de pătrate, la diferite niveluri ale arborelui. Condițiile while din pașii 4 și 11 și condițiile if de la pașii 8 și 13 presupun invocarea unei proceduri care verifică uniformitatea.**

1. Pentru /4 de la 0 la 4'—1 face: {Initialize}

ÎNCEPE.

2. Aflați coordonatele x și ^ din ecuațiile (6.4).
3. Plasați x.^ și 2""^ în lista *L.*

Sfârşit.

4. **În timp ce** se găsesc regiuni care pot fi îmbinate, **faceți:** {Merge}

ÎNCEPE.

5. Setați indexul / pentru a indica primul element al listei *L.*
6. În timp ce sfârșitul listei *L* nu a fost atins, faceți:

ÎNCEPE.

7. **Dacă** cele patru regiuni indicate de *j, j } =next(j), j z = următorulj }).* și *j3 = nexi(j 2)* au aceeași dimensiune, **atunci faceți:**

ÎNCEPE.

8. **Daca** unirea lor este uniforma conform cri terion utilizat, apoi *îmbinați* -le prin înlocuirea în *L a* dimensiunii primei regiuni cu 2J și prin deconectarea celorlalte trei.
9. Înlocuiți *j* cu *următorul (j 3).*

Sfârşit.

10. În caz contrar, înlocuiți *j* cu *next(j}.*

Sfârşit.

Sfârşit.

11. **În timp ce** se găsesc regiuni neuniforme **, faceți:** {Split}

ÎNCEPE.

12. **Pentru** toate elementele listei *L* **faceți:**

ÎNCEPE.

13. **[dacă** regiunea este uniformă, **atunci** avansați de-a lungul listei.

14. **Altfel,** înlocuiește-l cu cei patru copii ai săi.

Sfârşit.

Sfârşit.

15. **Sfârșitul algoritmului.**

X = ^d $_K$ *modulo2}'I* ^{K} ~^ ^{k}, (63a) A

y = SI-vJ' ^{2} "''* • «^ ^{b} >

Simbolul [xj indică cel mai mare număr întreg mai mic sau egal cu *x.* Domeniul coordonatelor este de la 0 la nL Pentru exemplul din figura 6.4. n - q - 3. Atunci indicele 213 îi corespunde

x - II + 12 + 04-3 și/ = II + 0-2 +14-5.

Cititorul poate verifica dacă caseta cu eticheta 213 pe acea figură se află într-adevăr pe a patra coloană și pe al șaselea rând. (Adăugăm una la valorile lui x și *y* , deoarece acestea sunt zero pentru primul rând sau prima coloană.) Astfel, indicele unei intrări din *L* indică către o regiune pătrată cu colțuri diagonal opuse la (x^) și (x+Z^'^+l""). Presupunem că este disponibil un criteriu de uniformitate pentru a decide dacă un grup de regiuni trebuie să fie îmbinat sau o singură regiune trebuie să fie împărțită în secțiune. 4.4.) Algoritmul scanează regiunea în așa fel încât atunci când unul dintre ele este divizat, primul său copil este examinat imediat pentru uniformitate.

Algoritmul de mai sus poate fi utilizat pentru rezolvarea altor probleme în afară de segmentarea imaginii. Secțiunea 17.2 descrie utilizarea sa pentru rezolvarea unei probleme de vizibilitate, dar fără fuziuni (vezi și Problema 6.9).

6.7 CODĂRILE LINIILOR ȘI GRAFUL DE ADJACENȚĂ A LINIILOR (LAC)

Când o imagine nu poate încadra în memoria principală, este adesea examinată de-a lungul liniilor paralele cu o direcție dată (scanare raster). Aceeași formă de traversare a imaginii este, de asemenea, comună ori de câte ori este utilizată tehnologia televiziunii. Dacă imaginea este de clasa 2, poate fi codificată ca o secvență de lungimi de o anumită culoare sau luminozitate. Este mai eficient să stocați astfel de informații doar pentru primul interval al fiecărei linii și să stocați diferențele de lungime și luminozitate (sau codul de culoare) pentru restul. O astfel de reprezentare este adesea numită *codificarea lungimii de rulare* (RLE), deși termenul a fost folosit inițial pentru imaginile pe două niveluri în care trebuiau stocate doar informații despre lungime. Imaginile din clasa I pot fi, de asemenea, codificate în acest mod, prin aproximarea intervalelor de niveluri de luminozitate cu media lor. Dacă numărul de intervale de pe o linie este /. atunci spațiul necesar pentru stocarea lungimilor și luminozității pe linie va fi

I(n+m), unde *n* și m au aceeași semnificație ca în secțiunea anterioară. Pentru valoarea tipică de opt, economii vor avea loc dacă / este mai mic de jumătate din numărul de pixeli de pe linie. Prin urmare, codificarea lungimii de rulare poate duce la economii semnificative de stocare și merită să implementați algoritmi pentru imagine.

analize sau grafice direct asupra datelor în această formă fără a reconstrui imaginea.t

Descrierea multor algoritmi de procesare a imaginilor folosind RLE devine mai ușoară dacă folosim un *grafic de adiacență a liniilor* (LAG). Nodurile unui astfel de grafic corespund intervalelor, iar ramurile conectează nodurile dacă intervalele corespunzătoare sunt pe linii adiacente, proiecțiile lor de-a lungul direcției de scanare se suprapun, iar pixelii lor au aceeași

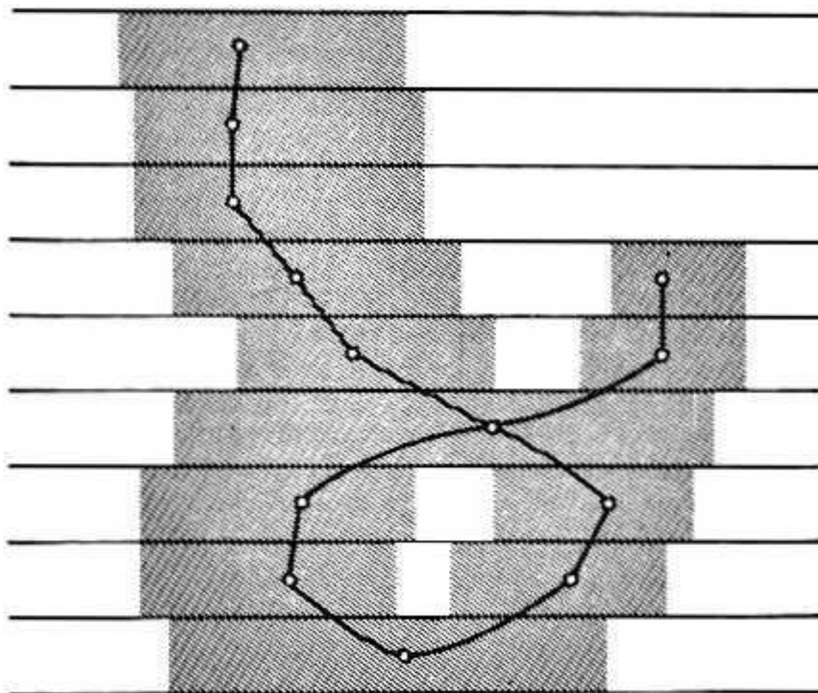


Figure 6.10 **Example of a line adjacency graph**

culoare. Figura 6.10 prezintă un exemplu de LAG. Dacă x_t indică coordonatele - punctelor finale ale intervalelor, atunci intervalele |x ‡-|.r $_t$) și [xi-i^) vor avea proiecții suprapuse dacă sunt valabile ambele inegalități următoare:*

$$x*-, < x, \text{ and } x\^{} < x_k . \qquad \textbf{(6.4)}$$

t Există chiar și dispozitive de afișare care utilizează RLE în memoria lor de reîmprospătare.

‡Amintiți-vă că (a .6) este un interval semiînchis, (dacă punctele satisfac inegalitatea *a* **Sx** <*b.*

Inegalitățile stricte implică faptul că intervalele care se ating doar la un colț nu sunt considerate suprapuse. Această restricție poate fi înlăturată pur și simplu făcând ca inegalitățile să nu fie stricte. Deoarece este ușor să verificați condițiile de suprapunere, nu este necesar să stocați în mod explicit graficul de adiacentă a liniilor.

LAG-ul poate fi dirijat avand ramuri indreptate de la nodul /I catre nodul *B* daca intervalul corespunzator nodului /I se afla deasupra intervalului corespunzator nodului *B.* Atunci se poate vorbi despre gradul *de mai sus* al unui interval-nod (numarul de intervale care il ating pe linia de sus) si gradul *de dedesubt* al aceluiasi {numarul de intervale care se ating pe linia de mai jos). Vom folosi notația *{a,b)* pentru a desemna gradele unui nod cu *a* pentru *deasupra* și *b* pentru *dedesubt.* Următoarele proprietăți pot fi dovedite cu ușurință.

Propunerea 6.2: Dacă un nod al LAG are gradul (0,^). atunci îi corespunde un maxim local pe direcţie verticală. Dacă gradul este (</.O), atunci acesta corespunde unui minim local în direcția verticală . □

Următorul rezultat, relaționând LAG-urile pentru diferite culori, este util în unele aplicații. Se presupune că există doar două culori, *X* și *Y,* în imagine și că LAG pentru culoarea *X* este construit folosind ecuația (6.4). în timp ce LAG pentru culoarea *Y* este construit folosind aceeași ecuație după relaxarea inegalităților stricte pentru a permite egalitatea, astfel încât intervalele care se ating doar la un colț să corespundă nodurilor conectate. (Raționamentul din spatele unei astfel de ipoteze este explicat în Secțiunea 7.2.) Figura 6.11 ilustrează construcția.

Propunerea 6.3: Dacă un nod al **LAG** pentru culoarea *X* are grad (mn) cu n>l atunci va exista un nod al LAG pentru culoarea *Y* cu gradul (0,d) în linia de mai jos. În mod similar, dacă m > I va exista un nod al LAG pentru culoarea F cu gradul (rf.O) deasupra acesteia. Inversul este valabil și în ambele cazuri. □

Dovada: Următoarea este o dovadă a părții care se ocupă cu gradul inferior mai mare decât unu. Fie x_M și x^+i punctele finale ale intervalului de culoare *X* care corespunde unui nod cu un grad inferior mai mare decât unu (sec Figura 6.11a). Aceasta înseamnă că în linia de mai jos există puncte finale de interval x_2 j-2. ^{x} 2.;-b ^{x} *2j*< 2nd x_2 j $_+$, cu proprietatea ecuației (6.5), de mai jos.

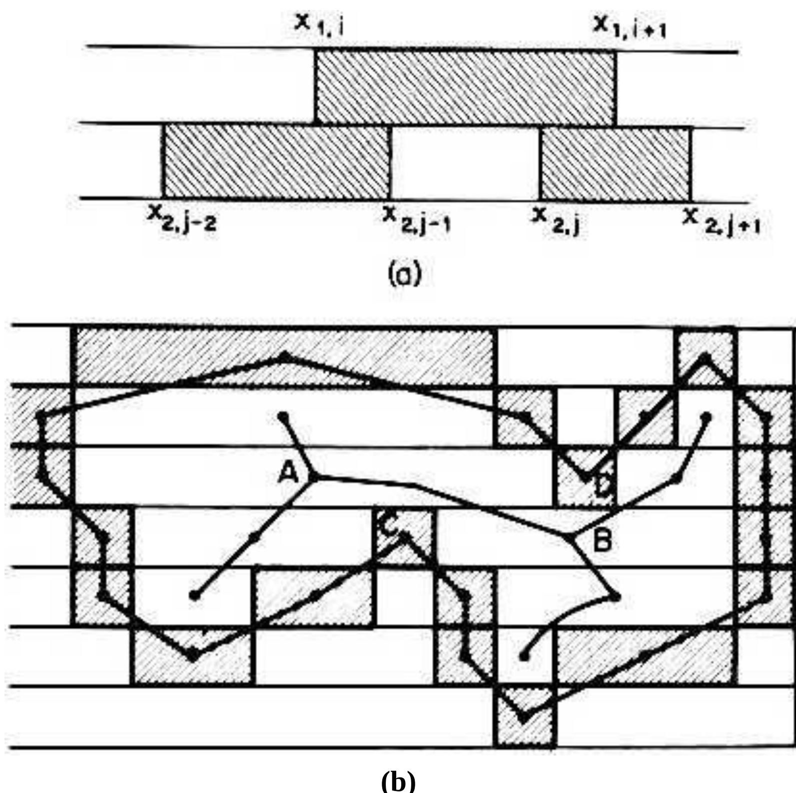


Figura 6.11 **Demonstrarea Propoziției 6.3: (a) notații utilizate în demonstrație, (b) configurație globală.**

*u < *2J-l și *x 2J* < x u+ | (6.5)

cu XJJ-I fiind punctul de capăt drept și *x 2 j* punctul de capăt din stânga intervalelor cu culoarea *X.* (Figura 6.11a.) Prin urmare, intervalul *x 1 j*-\x*X li* trebuie să aibă culoarea *Y.* Ar fi conectat la intervale de aceeași culoare pe linia de mai sus dacă și numai dacă este valabilă una dintre următoarele inegalități:

x 2^-i ^ x u sau ^u+t - *2./ • (6,6)

Dar mulțimea de inegalități din ecuația (6.6) este negația calităților ine ale ecuației (6.5) și de aceea nodul corespunzător intervalului X 2 J_|,X 2 J (de culoare F) trebuie să aibă peste gradul zero. (Aici folosim ipoteza despre un tip diferit de conectivitate pentru fiecare culoare: negarea unei inegalități nestricte este o inegalitate strictă.)

În schimb, să presupunem că intervalul x 2 jb*2j are peste gradul zero. Aceasta înseamnă că mulțimea de inegalități din ecuația (6.6) nu poate fi adevărată. Prin urmare, mulțimea inegalităților din ecuația (6.5) este adevărată și va exista un interval

de culoare *X* cu gradul inferior mai mare decât unu. Un argument similar poate fi formulat și în cazul situațiilor în care gradul de mai sus este mai mare decât unu. □

Propoziția 6.3 implică că intervalele în care gradul *de deasupra* sau *dedesubt* pentru culoarea *X* depășește unul corespund locurilor în care regiunea culorii *Y* are un maxim în direcția verticală cu culoarea *X* deasupra ei (cum ar fi nodurile A și C în Figura 6.11b), sau un minim în direcția verticală cu culoarea *X* dedesubt (cum ar fi nodurile B și D în Figura 6.1 lb).

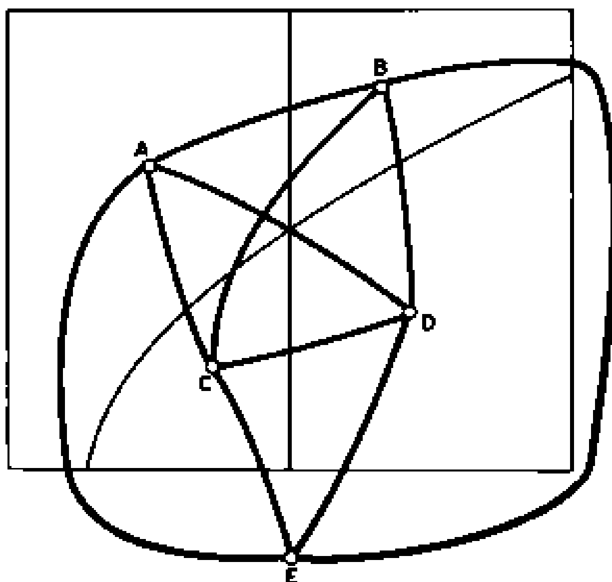


Figura 6.12 Restricții ale definiției adiacenței pentru regiuni. Imaginea conține cinci regiuni și este clar că perechile (,<.«), *{BD), (DC)* și *(CA)* împărtășesc o limită și că nodurile respective ale RAG trebuie să fie legate. De asemenea , *A, B,* C și *D* sunt toate legate de £. [dacă legăm perechile *(AD)* și (5.C), atunci RAG devine neplanar. O mică perturbare în punctul în care cele patru regiuni se întâlnesc va face ca doar una dintre perechi să fie adiacentă, iar RAG-ul rezultat va fi plan.

6.8 CODĂRILE REGIONALE ȘI GRAFUL DE ADJACENȚĂ A REGIUNII (RAG)

Această formă este potrivită în special pentru imagini de clasa 2 sau aproximări ale imaginilor din clasa I. O regiune conectată de o anumită culoare poate fi descrisă complet prin conturul său, iar aceasta ocupă de obicei mult mai puțin spațiu decât o descriere în termeni de pixeli. Operațiunile peste astfel de imagini pot fi facilitate dacă este disponibil graficul de adiacență a regiunii. Nodurile acestui graf corespund unor regiuni, iar două noduri sunt unite printr-o ramură dacă regiunile respective sunt adiacente. Ar trebui să subliniem că conceptul de adiacență trebuie definit cu atenție

pentru imaginile discrete și îl vom discuta în detaliu mai târziu (Capitolul 7). Aici subliniem doar că, indiferent de definiția care va fi utilizată, aceasta nu ar trebui să permită conexiuni care vor face RAG-ul neplanar. În special, când patru regiuni se întâlnesc într-un colț, nu ar trebui să permită ambele conexiuni diagonale, așa cum se arată în Figura 6.12. Figura 6.13 prezintă un exemplu de grafic suprapus pe o imagine. Este posibil să extrageți informații semnificative despre imagine prin verificarea diferitelor proprietăți ale graficului.

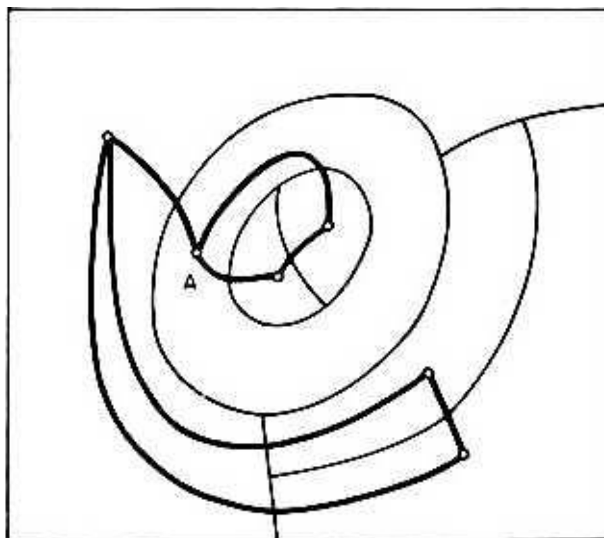


Figura 6.13 Un grafic de adiacență a regiunii

Mai jos este o listă cu câteva caracteristici ale RAG și proprietățile corespunzătoare ale imaginii.

(a) Gradul unui nod corespunde numărului de regiuni adiacente unei anumite regiuni. Acest număr este de obicei proporțional cu dimensiunea regiunii și, prin urmare, nodurile de grad înalt vor corespunde regiunilor mari.

(b) Dacă o regiune înconjoară complet alte regiuni (A în Figura 6.13), atunci nodul corespunzător este un nod *tăiat* în RAG. Dacă toate regiunile dintr-o imagine sunt fără găuri (adică, sunt pur și simplu conectate), atunci niciuna dintre ele nu va înconjura pe alta și graficul nu va avea nicio tăietură. Deoarece există algoritmi eficienți (sec. Note bibliografice) pentru găsirea nodurilor tăiate ale unui grafic, este posibil să se obțină toate informațiile topologice despre imagine din RAG.

Nodurile RAG pot fi etichetate cu informații despre regiuni, astfel încât alte proprietăți, în afară de topologie, pot fi investigate. RAG poate fi folosit și pentru o formă sofisticată de paginare. Paginile sunt formate din părți ale unei imagini care corespund unor grupuri mici de noduri (sau chiar unui singur nod), iar acest lucru oferă o subdiviziune mai naturală decât un sistem de grilă arbitrar.

În cele din urmă, RAG oferă structura de bază a datelor pentru segmentarea în funcție de creșterea regiunilor, inclusiv algoritmi de divizare și îmbinare. Un dezavantaj al algoritmului 6.6 este că, la sfârșit, regiunile cu caracteristici similare nu pot fi îmbinate dacă nu au un strămoș comun în arborele quad. Introducerea RAG în această ultimă etapă permite gruparea unor astfel de regiuni.

6.9 REPREZENTARI ICONICE

Reprezentările iconice sunt destul de complexe, dar sunt cel mai eficient mod de stocare și manipulare a imaginilor. Se presupune că o imagine este subîmpărțită în subimagini care pot fi pagini arbitrare (ca în secțiunea 6.3), pătrate ale unui arbore quad, regiuni rezultate dintr-o segmentare . etc. Cu toate acestea, în loc să stocheze toate imaginile, unele dintre ele sunt înlocuite cu o etichetă descriptivă. Exemplul 1.1 ilustrează o formă de reprezentare iconică. De regulă, trebuie utilizată o anumită formă de recunoaștere a modelelor înainte de a obține această formă.

6.10 STRUCTURI DE DATE PENTRU AFIȘARE

Structurile de date descrise mai sus sunt utile pentru aplicații

unde se ocupă de imagini existente. În grafică se începe cu descrieri ale imaginilor, iar acestea pot fi într-o formă nepictorială. În general, astfel de descrieri vin sub forma unor funcții polinomiale pe bucăți ale unei variabile (pentru curbe) sau a două variabile (pentru suprafețe). În grafica vectorială, unde comenzile primitive afișează doar linii drepte, funcțiile sunt liniare pe bucăți, formând poligoane sau poliedre. În ambele cazuri, descrierea obiectului este un grafic cu ramuri care sunt segmente de linie dreaptă. Putem folosi termenul *de graf obiect* pentru această structură de date. Dacă descrierea implică segmente curbilinie sau pete de suprafață de ordin înalt, atunci putem avea totuși un grafic obiect, dar cu *ramuri etichetate:* etichetele sunt coeficienții curbei sau suprafeței. Alternativ , se poate folosi un grafic de adiacentă a regiunii (neplanare) ale cărui noduri vor corespunde unor petice de suprafață, iar dacă peticele sunt adiacente , atunci ramurile vor conecta nodurile respective. Un astfel de „grafic de adiacență a patch-urilor" este adecvat pentru graficele raster.

În principiu, este ușor să construiți afișaje din astfel de grafice. O provocare practică este crearea unor structuri de date care să ajute la rezolvarea diferitelor probleme întâlnite la generarea afișajului. Pot fi necesare presortarea și introducerea de link-uri suplimentare pentru a accelera soluționarea problemelor de vizibilitate, de exemplu. Vom reveni la aceste întrebări în capitolul 17.

6.11 NOTE BIBLIOGRAFICE

Problemele traversării graficelor și structurile de date aferente sunt discutate în majoritatea cărților generale despre algoritmi sau structuri de date (de exemplu, [6.AHU], (6.KN]). Vezi și [3.PA] pentru un tratament în contextul recunoașterii modelelor. Vezi (6.RO] și [6.DEL] pentru o discuție despre problema păstrării stocării geometrice în adiacenta liniară.

Arborele quad este una dintre cele mai comune structuri de date discutate în literatură. Prima sa aplicație binecunoscută a fost în contextul unui algoritm de eliminare a liniilor ascunse de Warnock [6.WA). La începutul anilor șaptezeci, a găsit o utilizare pe scară largă în recunoașterea modelelor și procesarea imaginilor ([6.HP], [6.HS], [6.KD], [6.TP], [6.TA]). Utilizarea arborelui quad pentru afișajele cu informații brute a fost descrisă mai întâi de Sloan și Tanimoto (6.STJ. Arborele binar a fost propus recent de Knowlton (6.KK] (Figura 6.8 este din [6.KK]). Arborii quad sunt acum un subiect de cercetare popular și proprietățile lor sunt investigate pe larg. Sec.]6.SR] pentru o revizuire).

Graficele de adiacență de linii au fost, de asemenea, utilizate pe scară largă, deși utilizarea este adesea implicită. Algoritmi care efectuează operații asupra unei imagini

sunt descrise în termeni de parcurgere a pixelilor și a intervalului fără referire la grafic. [6.GR] este probabil cea mai veche lucrare în care este utilizat LAG (implicit). Consultați [3.PA] pentru mai multe exemple de utilizare a acestuia în recunoașterea modelelor și procesarea imaginilor. [6.SH] descrie o utilizare explicită în grafică (a se vedea capitolul 8 pentru mai multe despre acest subiect).

Graficul de adiacență a regiunii este probabil structura de date mai veche și mai evidentă pentru procesarea imaginilor și cea mai mare parte a literaturii despre analiza scenei folosește diferite versiuni ale acesteia.

6.12 LITERATURA RELEVANTA

[6.AHU] Aho, AV; Hopcroft, JE; și Ullman, JD *The Design and Analysis of Computer Algorithms,* **Reading, Mass.: Addison-Wesley, 1974.**

[6.DEL] DeMillo, RA; Eisenstat, SC; şi Lipton, RJ „Preserving Average Proximity in Arrays"

CACM, **21** **(1978), pp. 228-231.**

[6.GR] Grimsdale, RL; Vara, FH; Tunis. CJ; și Kilburn, T. „Un sistem pentru recunoașterea automată a modelelor”, *Proc. 1EE.* **106B (1959), p. 210-221.**

[6.HP] Horowitz, SL și Pavlidis, T. „Picture Segmentation by a Tree Traversal Algorithm”, *Journal of the ACM,* **23 (1976), pp. 368-388.**

[6.HS] Vânător. GM și Steiglitz, K. „Operations on Images Using Quad Trees”, *IEEE Trans. Pattern Analysis and Machine Intelligence,* **PAMI-1 (1979), pp. 145-153.**

[6.KD] Klinger, A. și Dyer, CR „Experiments on Picture Representation Using Regular Decomposition”, *CGIP,* **5 (1976), pp. 68-105.**

[6.KK] Knowlton, K. „Progressive Transmission of Gray Scale and B/W Pictures by Simple, Efficient and Lossless Encoding Schemes,” *IEEE Proceedings.* **68 (1980), p. 885-896.**

[6.KN] Knuth, DE, *Fundamental Algorithms,* **voi. 1. Reading, Mass.: Addison-Wesley, 1973.**

I6.RO] Rosenberg, AL "Preserving Proximity in Arrays", *SIAM J. Computing.* **4 (1975), p. 443-460.**

[6.SH] Shani, U. „Filling Regions in Binary Raster Images — a Graph-theoretic Approach”, *SIGGRAPH'80,* **pp. 321-327.**

(6.SR) Sammet, H. și Rosenfeld, A. „Quadtree Representations of Binary Images”, *Fifth Intern, on Pattern Recognition* **, decembrie 1980, p. 815-818 (Publicat de IEEE Computer Society. IEEE Nr. 80CH.**

[6.ST] Sloan, KR, Jr. și Tanimoto, SL „Progressive Refinement of Raster Images,” *IEEE Trans, on Computers.* **C-28 (1979), p. 871-**

(6.TAJ Tanimoto, SL, „Hierarhical Approaches to Picture Processing,” *Ph. D. Dissertation.* **Dept, of Electrical Engineering, Princeton University, August 1975, 241pp.** *(Disponibil de la University Microfilms).*

(6.TP] Tanimoto, SL și Pavlidis, T. „A Hierarchical Data Structure for Picture Processing,” *CGIP.* **2 (1975), pp. 104-119.**

[6.WA] Warnock, JE „A Hidden-Surface Algorithm for Computer Generated Half-tone Pictures”, *Raport tehnic 4-15.* **Departamentul de informatică, Universitatea din Utah, 1969.**

6.13 PROBLEME

6.1. Un dezavantaj al utilizării tablourilor bidimensionale este că în unele limbaje de calculator dimensiunea lor trebuie specificată în timpul compilării . Pe de altă parte, trebuie fixat în avans doar dimensiunea maximă a unui tablou unidimensional. Scrieți un program folosind acest concept pentru stocarea pixelilor unei imagini și includeți proceduri pentru evaluarea coordonatelor *xy* din locația unui pixel în matrice. Implementați algoritmul 6.2 folosind o astfel de matrice. Definiți ca pixeli adiacenți *p* cei opt pixeli care pot fi atinși de la *p* printr-una din direcțiile din figura 1.2.

6.2. Arătați că se poate parcurge corect o componentă conectată a unui grafic chiar și atunci când un nod este plasat în stivă de mai multe ori.

6.3. Repetați problema 6.1 cu următoarea modificare în algoritmul 6.2: în loc să plasați pixelii adiacenți *p* (pixelul curent) pe stivă, plasați *p* însuși. Cum simplifică acest lucru algoritmul? Mai trebuie să utilizați un marcaj suplimentar pentru pixelii plasați în stivă? Plătiți vreun preț pentru această simplificare?

6.4. Scrieți un program de traversare a graficului care imprimă perechi de noduri conectate printr-o cale care conține doar noduri de gradul doi. (Puteți să vă gândiți la grafic ca la o rețea de comunicații, unde nodurile de gradul unu sunt *terminale,* nodurile de gradul mai mare decât doi sunt *interschimbări,* iar nodurile de gradul doi sunt *amplificatoare.* Programul dvs. va produce toate perechile de schimbătoare și terminale care au o conexiune directă, cu excepția amplificatoarelor. Pentru o problemă similară în procesarea imaginii, consultați capitolul 9).

6.5. Această problemă este legată de paginare. Luați în considerare un set de pătrate identice dispuse pe plan astfel încât acestea să formeze o matrice de pătrate /VXA f · În acest aranjament există 2/V(A'-l) perechi de pătrate care împart o latură. (Figura 6.14 arată cazul pentru *N* =4.)

1	2	3	4
5	6	7	8
9	10	II	12
13	14	15	16

Figarc 6.14 Aranjarea pătratelor pentru problema 6.5.

Acum ni se cere să aranjam pătratele într-o manieră liniară, astfel încât să păstrăm cât mai mult posibil proximitatea bidimensională. Examinați diverse strategii pentru a face acest lucru, precum și diferite măsuri de proximitate. De exemplu, dacă aranjam pătratele în ordine numerică (conform sistemului din figura 6.14), atunci adiacența directă va fi păstrată pentru N(/^-l) perechi. Un aranjament în zig-zag face același lucru pentru perechile (/Vl)(/V+l). Există o strategie care păstrează adiacența pentru mai multe perechi? Alte măsuri de proximitate includ distanța maximă dintre membrii perechilor care erau adiacente, distanța medie dintre astfel de perechi etc.

6.6. Modificați algoritmii 6.4a și 6.4 astfel încât, dacă copiii unui nod sunt toți la fel, ei nu sunt incluși în arborele quad.

6.7. Având în vedere coordonatele x și y (ale unui pixel în intervalul 0 la 2" — 1, găsiți eticheta acestuia conform sistemului ilustrat în Figura 6.4.

6.8. Costul major al unei operațiuni de împărțire sau îmbinare în algoritmul 6.6 este necesitatea de a verifica din nou pătratele rezultate pentru uniformitate. Utilizați analiza secțiunii 4.4 pentru a arăta că costul unei fuziuni nu poate depăși costul unei divizări.

6.9. Modificați algoritmul 6.6 astfel încât să găsească marginile tuturor regiunilor într-o imagine cu două niveluri.

ANEXA 6.A: INTRODUCERE ÎN GRAFICE

Teoria grafurilor este un subiect vechi cu o literatură vastă și numeroase aplicații. Se pare că cunoașterea unor concepte elementare ale subiectului poate ajuta o persoană să înțeleagă majoritatea algoritmilor în care teoria grafurilor este utilizată în procesarea informațiilor picturale. Scopul acestei anexe este de a introduce unele dintre cele mai comune concepte de grafice.

Un grafic este un set de **puncte** sau **noduri** conectate prin **linii** sau **ramuri.** Exemple de grafice sunt prezentate în Figurile 6.10. 6.11b. 6.12 și 6.13. **Gradul** unui nod este egal cu numărul de ramuri incidente cu acesta. Astfel, toate nodurile graficului din figura 6.12 au gradul patru în timp ce sunt în

Figura 6.13 gradul diferitelor noduri variază de la două la patru. Străzile unui oraș sunt un exemplu de grafic în care nodurile corespund intersecțiilor sau capetelor străzilor, iar ramurile corespund segmentelor de stradă dintre intersecții sau capete. În „graficul străzilor” majoritatea nodurilor au gradul patru, iar străzile fără fund se termină la

nodurile de gradul unu.

Un grafic direcționat este unul în care unele sau toate ramurile au o direcție asociată acestora. În exemplul străzii, graficul devine direcționat dacă luăm în considerare străzile cu sens unic. Într-un astfel de grafic putem vorbi despre gradul **de ieșire** și gradul **de intrare** al unui nod, numărând ramurile îndreptate spre el și ramurile îndreptate spre el. O variație a acestei terminologii este oferită în Secțiunea 6.7.

O cale de la nodul *A* la nodul £ este o succesiune de ramuri pe care trebuie să o parcurgem pentru a merge de la /I la 5. Un circuit este un drum al cărui început și sfârșit coincid. Se spune că un graf **este conectat** dacă există o cale între orice pereche de nodurile sale.

Un **arbore** este un graf conectat fără circuite.

Un subgraf al unui graf <7 este un graf care conține unele sau toate nodurile lui *G* și unele sau toate ramurile sale și nu alte noduri sau ramuri. **Un arbore de acoperire** al unui graf conectat 6 este un subgraf al lui *G* care conține toate nodurile lui *G* și suficiente ramuri pentru a-l face conectat fără a crea circuite.

O modalitate obișnuită de vizualizare a copacilor este să alegeți un nod ca **rădăcină** a arborelui, apoi să desenați toate nodurile conectate la acesta sub el și așa mai departe (vezi Figura 6.3). Nodurile conectate la *A.* nodul de deasupra lor, se numesc **copiii** lui /I, iar *A* se numește **părintele lor.**

Capitolul 7

IMAGINI BILLEVEL

7.1 INTRODUCERE

Acest capitol și următoarele două capitole se ocupă de imagini de clasa 2 în care regiunile de nivel de gri sau culoare sunt bine definite. O preocupare majoră în studiul unor astfel de imagini este conceptul de *formă,* termen care nu este ușor de definit cantitativ. O persoană se confruntă cu această problemă atunci când eșantionează o imagine analogică pe două niveluri. Dimensiunea celulelor grilei de eșantionare trebuie să fie suficient de mică, astfel încât formele regiunilor de o anumită culoare să rămână nealterate în reconstrucția imaginii. Secțiunile 7.2, 7.4 și 7.6 tratează acest lucru și alte aspecte ale problemei digitizării. Un alt set de probleme în procesarea unor astfel de imagini implică transformarea lor într-un set de curbe sau întoarcerea de la un set de curbe la regiuni plane. Trasarea conturului unei regiuni este discutată în Secțiunea 7.5. Reconstituirea regiunii de la conturul său, sau *umplerea acesteia,* va fi tratată în Capitolul 8. În loc să găsiți conturul, se poate alege să subțiați o regiune la scheletul său, o figură de tip chibrit ale cărei curbe sau linii reflectă forma regiunii. Algoritmii *de subțiere* vor fi tratați în Capitolul 9.

Algoritmii de trasare a contururilor, de umplere a contururilor și de subțiere sunt întâlniți în diverse aplicații, dar au multe caracteristici în comun. Toate implică parcurgerea unei regiuni plane și, în timp ce procesul este destul de simplu (de exemplu, folosind algoritmul 6.2), apar o serie de probleme subtile. Este ușor să dați definiții precise pentru trasarea contururilor, umplerea conturului și subțierea în plan analog, cu condiția ca

se ocupă de mulțimi mărginite care au contururi care satisfac anumite condiții de netezime. Este destul de dificil să dai definiții precise în plan discret. Dedicăm Secțiunea 7.3 unei discuții despre geometria discretă și revenim la subiectul din Secțiunile 7.6 și

7.7. În cele din urmă, deoarece analiza formei este o motivație majoră pentru trasarea conturului în imaginile de clasa 2, vom dedica Secțiunea 7.8 unei scurte discuții despre cum să obțin descrieri de forme din datele de contur.

7.2 PRELEVARE SI TOPOLOGIE

Eșantionarea imaginilor binivel sau, mai general, de clasa 2 prezintă o provocare specială, deoarece astfel de imagini implică funcții de pas. Transformarea Fourier a unor astfel de funcții este diferită de zero pentru toate frecvențele. Prin urmare, nu există un interval finit de eșantionare care să permită digitizarea fără erori. Intuitiv, acest lucru pare evident dacă se dorește să păstreze informații despre locația exactă a granițelor dintre regiuni. Problema poate fi văzută în cazul unidimensional din Figura 7.1. Dacă cineva este dispus să tolereze o eroare în locația granițelor, atunci este posibil să găsiți o soluție după cum urmează.

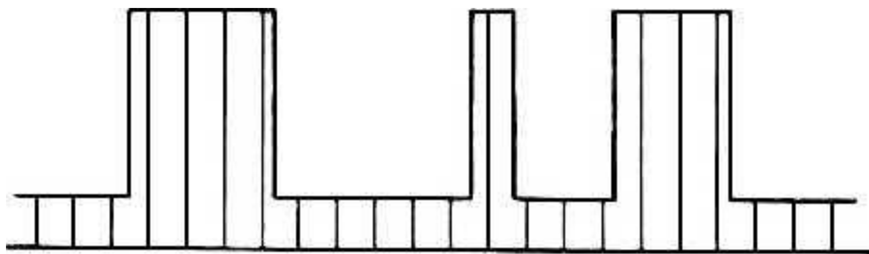


Figure 7.1 Sampling of a bilevel signal. Each peak will be
se păstrează în versiunea discretă dacă distanța dintre două eșantioane este mai mică decât lățimea minimă a unor astfel de vârfuri.

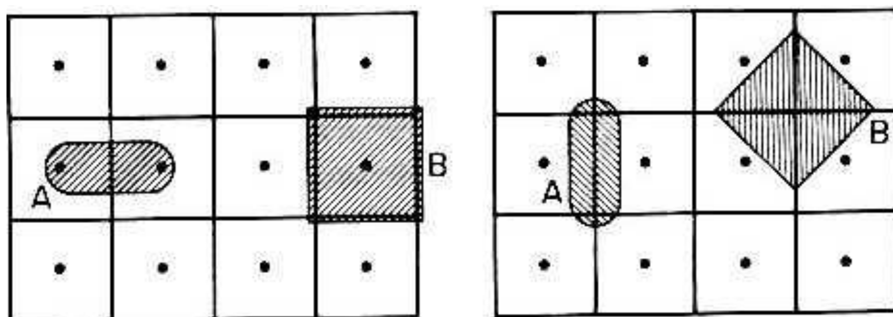
Datorită funcției de împrăștiere punctuală g(r) (secțiunea 2.3.2), eșantioanele nu trebuie să fie egale cu unul dintre cele două niveluri inițiale. Ele vor lua valori intermediare în apropierea graniței dintre niveluri. Pentru a recupera o reprezentare pe două niveluri, probele trebuie comparate cu un prag între niveluri. Aceasta va introduce o eroare în locația graniței nu mai mult de jumătate din lungimea intervalului în care $g(r)$ este diferit de zero. Presupunem că acest interval este foarte mic în comparație cu intervalul de eșantionare, așa cum trebuie să fie în orice digitizator bine proiectat.

Fie h mărimea toleranței. Dacă intervalul de prelevare este mai mic

decât h, atunci obținem precizia dorită. Niciun interval nu se va pierde dacă lățimea minimă depășește h. Se poate arăta că, dacă funcția cu două niveluri modifică valorile numai în anumite puncte care sunt situate la un multiplu întreg al lui h , atunci transformarea sa Fourier va fi zero pentru frecvențe înalte și funcția poate fi eșantionată corect. Chiar dacă această ipoteză nu este realistă, ipoteza lățimii minime este și cu cât h este mai mic în comparație cu acea lățime, cu atât eroarea introdusă de eșantionare

este mai mică. În special, putem evita *aliasarea.* ceea ce pentru imaginile cu două niveluri înseamnă pierderea golurilor astfel încât funcția să pară uniformă pe un interval când de fapt nu este.

O altă abordare a eșantionării semnalelor pe două niveluri implică conversia acestora în scară de gri. Un filtru care înlocuiește valoarea într-un punct cu media vecinilor săi este un filtru trece-jos și reduce transformarea Fourier la valori aproape de zero la frecvențe înalte. Prin urmare, imaginea rezultată în scala de gri poate fi eșantionată fără erori, cu excepția celor introduse de mediere. Această abordare nu este întotdeauna fezabilă pentru imagini, deoarece există aplicații (de exemplu, fotocompunere) în care ieșirea este restricționată la o formă pe două niveluri. Dacă eșantioanele sunt pragizate astfel încât semnalul rezultat să fie binivel, atunci introducem erori în localizarea limitelor și această metodă este echivalentă cu prima.



Figare 7.2 Ilustrarea dificultăților de stabilire a criteriilor de eșantionare pentru imagini. Regiunile J și *B* sunt eliminate în figura din stânga, dar lipsesc în figura din dreapta.

Translația problemei de eșantionare (și soluția ei) în două dimensiuni este netrivială. Deși există o singură formă posibilă de eșantionare de-a lungul unei linii (subdiviziune în intervale), numărul de posibilități în două dimensiuni este infinit! Trebuie specificată nu numai dimensiunea grilei de eșantionare, ci și forma acesteia. Totuși, ne limităm discuția la grile pătrate, din motivele discutate în Secțiunea 2.3.2. Dacă latura pătratului este *h,* atunci este evident că nicio regiune nu va fi pierdută dacă toate

acestea sunt suficient de mari pentru ca un pătrat al grilei să poată fi înscris în cea mai mică regiune. Acest lucru va fi adevărat dacă valoarea eșantionului este determinată de centrul celulei sau de o medie pe întreaga sa zonă. Din păcate, această caracterizare este atât dependentă de translație, cât și de orientare, așa cum se poate vedea din Figura 7.2. În plus, nu spune nimic despre păstrarea topologiei regiunilor. Niciuna dintre aceste probleme nu a fost prezentă în cazul unidimensional!

Din punct de vedere practic, principala preocupare în eșantionarea imaginilor din clasa 2 este păstrarea formei. Păstrarea formei necesită ca seturile conectate din imaginea analogică să fie seturi conectate în imaginea discretă, precum și să aibă o serie de alte proprietăți. Preocuparea cu privire la conectivitate duce imediat la întrebări topologice și de aici încep dificultățile. Este practic imposibil să existe o definiție intuitivă a topologiei asupra unei mulțimi ale cărei elemente sunt, într-un fel, izolate unele de altele.

Cu toate acestea, există o cale de ieșire. În loc să luăm în considerare transformarea dintr-o imagine analogică într-un set de pixeli discreti, vom lua în -considerare transformarea dintre imaginea analogică și reconstrucția ei , de exemplu, imaginile analogice obținute prin umplerea celulelor afișate cu culoarea pixelului aflat în celulă. Acum trebuie să ne ocupăm de relațiile dintre imaginile care sunt ambele analogice. Majoritatea dificultăților dispar. Implicațiile practice ale abordării nu sunt rele. Când oamenii vorbesc despre imagini discrete, de obicei au în minte imaginile reconstruite. În special, să fiu *eu* imaginea originală. În timpul eșantionării, fiecărui pixel i se dă culoarea centrului celulei de eșantionare corespunzătoare. O nouă imagine *J* poate fi apoi creată prin umplerea fiecărei celule de afișare cu culoarea pixelului corespunzător. Figurile 7.3a și 7.3b prezintă acest proces.

Înainte de a discuta relațiile topologice dintre cele două imagini este necesar să se clarifice termenii folosiți. Se spune că două mulțimi *A* și *B* sunt echivalente din punct de vedere topologic dacă există o mapare unu-la-unu între *A* și *B y* și dacă atât maparea de la /I la *B,* cât și inversa sa de la *B* la *A,* sunt continue ([7.KE], p. 87). În procesarea imaginilor, de obicei se ocupă de topologia indusă de metrica euclidiană și, prin urmare, echivalența topologică înseamnă că există o mapare unu-la-unu, astfel încât punctele care sunt apropiate unul de celălalt sunt mapate la puncte care sunt apropiate* și invers. Informal, asta înseamnă că se poate

t Vom explica pe scurt ce înțelegem prin *topologie* pentru cititorii care nu sunt familiarizați cu termenul.

Cititorii care simt că termenul „aproape” nu este suficient de precis ar trebui să consulte un text despre analiză pentru definirea funcțiilor continue.

transformați un set în altul prin întindere și strângere fără a rupe sau tăia nimic. Astfel, dacă o regiune are o gaură nu poate fi echivalentă topologic cu o regiune fără gaură, deoarece o astfel de transformare ar necesita tăiere.

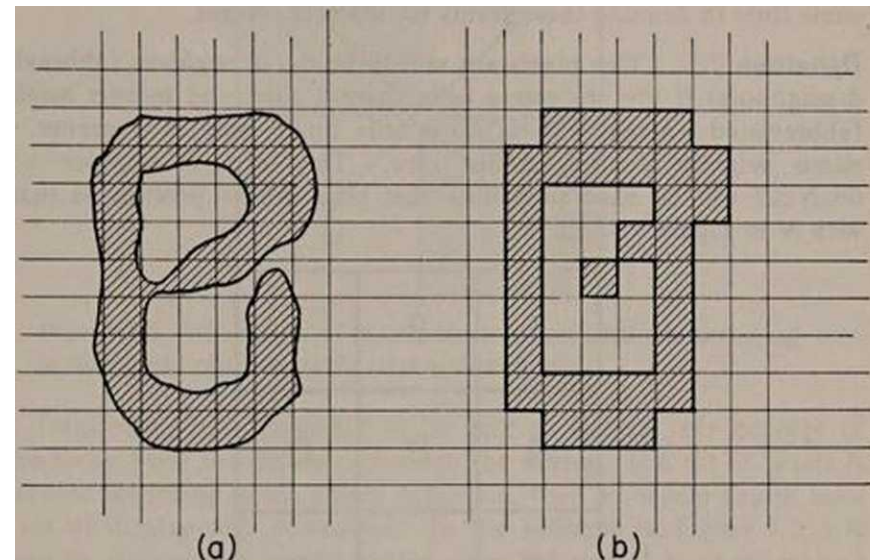


Figura 7.3 (a) Imagini originale și (b) reconstruite după eșantionarea cu o grilă pătrată

Nu există nicio dificultate în definirea conceptelor topologice pe oricare dintre imaginile prezentate în Figura 7.3: ambele sunt pe planul continuu unde o metrică euclidiană este bine definită. Mai mult, se poate arăta că ele sunt într-adevăr echivalente din punct de vedere topologic. A demonstra acest lucru cu rigurozitate ne-ar duce dincolo de sfera acestui text, dar se poate fi de acord, fără a recurge la niciun formalism, că, de exemplu, ambele seturi întunecate sunt conectate și că fiecare are o gaură. Acest exemplu arată, de asemenea, că păstrarea topologiei ar putea fi o condiție necesară, dar că nu este suficientă pentru păstrarea formei. În mod clar, rezultatul din figura 7.3b este nedorit din acest punct de vedere. Chiar dacă solicitarea păstrării numărului de regiuni conectate de toate culorile nu este suficientă, este cel puțin un început.

Datorită simplității criteriului de eșantionare unidimensională, cineva poate fi tentat să solicite ca constrângerea unidimensională să fie satisfăcută de-a lungul tuturor liniilor de scanare. Din păcate, acest criteriu este probabil să eșueze de-a lungul liniilor care sunt aproape tangente la granițele regiunilor. Este necesar să atacăm problema direct și vom face acest lucru în Secțiunea 7.4. Mai întâi trebuie să divagam pentru a introduce anumite concepte simple din geometria discretă.

7.3 ELEMENTE DE GEOMETRIE DISCRETA

După cum am menționat în introducere, multe concepte geometrice care sunt bine definite pentru imaginile analogice nu au corespondente în imaginile discrete (seturi de pixeli). Prin urmare, este necesar să se dedice ceva timp definirii acestor termeni pentru imaginile discrete.

Definiția 7.1: Se spune că doi pixeli sunt *vecini direcți* (abreviați d-neighbors) dacă celulele respective au o latură, și *vecini indirecti* (abreviați i-neighbors) dacă acele tavane se ating doar la un colț. Numele *vecin* denotă oricare dintre tipurile. Termenul *N-vecin* unde *Q<N^1* va fi folosit pentru a desemna acel pixel a cărui poziţie este marcată cu TV în Figura 7.4. O

3	2	I
4	P	O
5	6	7

Figura 7,4 Notație utilizată pentru definirea locațiilor relative ale pixelilor în raport cu un pixel *P.*

Rețineți că vecinii d sunt /V-vecini cu *N* par, în timp ce vecinii i- corespund cu *N* impar.

Definiția 7.2: O *cale i* (sau pur și simplu *cale)* este o secvență de pixeli ^ 1(4], ■ • • , *A„* astfel încât pentru £>!, 4*_| este un vecin al lui *A k* și pentru k<n, 4* + i este un vecin al lui /I* Termenul d-cale se referă la o secvență similară, dar în loc să fie pixeli A-ne sunt vecini simpli . unul în care toți pixelii sunt diferiți și niciun pixel are mai mult de doi d-vecini în cale. O ***cale închisă*** este una în care primul și ultimul pixel coincid

Definiția 7.3: Un set de pixeli *S* este *conectat* (sau *i-conness)* dacă pentru fiecare pereche de pixeli *C* și D din S, există o cale i ale cărei prim și ultimul element sunt *C* și respectiv *D* și toți ceilalți pixeli ai săi aparțin lui *S.* Termenul d-conectat are sensul evident. □

Problema conectivității a fost subiectul unora

atenție în literatură din cauza celor două definiții posibile. În special oamenii au încercat să rezolve paradoxul ilustrat în Figura 7.5.

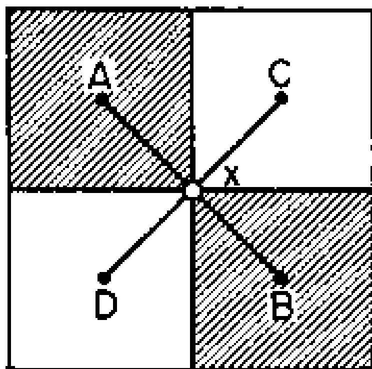


Figura 7.5 Ilustrarea contradicțiilor topologice care apar în definițiile conectivității pe o rețea discretă

Intuitiv, este de dorit să se poată transfera conceptul de conectivitate din planul discret în cel analog. Dacă un set de pixeli este conectat conform definiției de mai sus, atunci am dori să avem setul de celule de afișare conectat. În exemplul din figura 7.5 este firesc să presupunem că pe planul analog seturile de puncte negre și puncte albe sunt disjunse și că cele două împreună acoperă partea din plan prezentată, deoarece nu există puncte de altă culoare. Dacă definim conectivitatea ca i-conectivitate, atunci avem o situație în care o cale de la X la 5 (vezi Figura 7.5) poate intersecta o cale de la *C* la *D,* chiar dacă ambele se află în întregime în mulțimi disjunse. Astfel, punctul *X* trebuie să aparțină atât setului alb, cât și celui negru. dacă presupunem d-conectivitate, atunci nici setul de pixeli negri și nici setul de pixeli albi nu este conectat. Aceasta înseamnă că o cale de la /I trudă (sau *C* la *D)* nu poate fi niciodată în întregime într-un singur set. Deoarece segmentele de dreaptă *AB* și *CD* se încrucișează în punctul *X,* acest punct nu trebuie să aparțină niciunei mulțimi. Dar acest lucru contrazice presupunerea că celulele albe și negre acoperă complet regiunea plană prezentată în Figura 7.5.

Paradoxul apare din cauza neglijenței în stabilirea corespondenței dintre seturile de pixeli și seturile de celule. În plan analog, seturile pot să conțină sau nu limitele lor,! iar în prezent

t Granița unei mulțimi 5 este definită formal ca mulțimea tuturor punctelor din plan cu următoarea proprietate. Dacă *P* aparține graniței lui *S,* atunci toate cercurile centrate pe *P* includ atât punctele lui *S* , cât și ale complementului său, indiferent cât de mică este raza lor. *P* mai. sau poate să nu aparțină lui *S.*

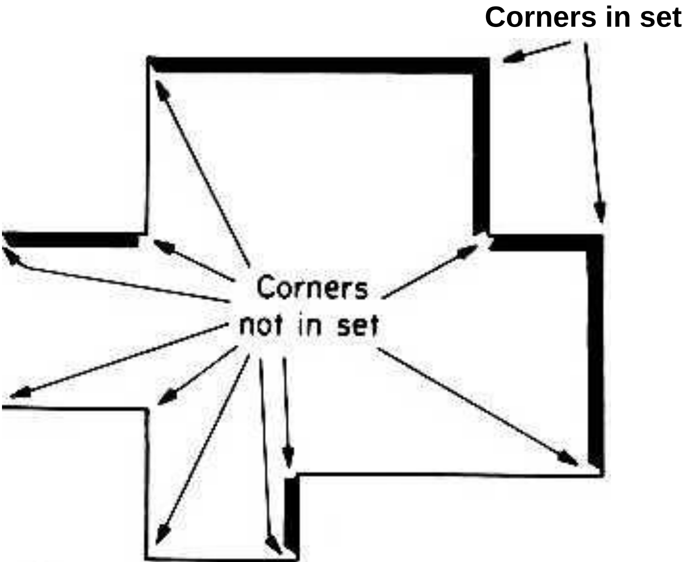
de exemplu, includerea unor astfel de granițe determină răspunsul la întrebarea conectivității. Avem o serie de opțiuni: una dintre ele este să presupunem că toate celulele negre își conțin limitele (adică, sunt *închise* în sens topologic), în timp ce celulele albe nu (adică sunt *deschise).* Apoi setul de celule negre este conectat, iar setul de celule albe nu. Acest lucru poate fi văzut clar dacă definim regiunile din Figura 7.5 analitic după introducerea unui sistem de coordonate *xy* . Dacă *Z* este dimensiunea laturii pătratelor, iar originea este în colțul din stânga jos, avem:

0<x<Z *Z^y^2Z*
0<y<Z Z<x^2Z

Square A (black)
Square B (black)
Square C (white)
Square D (white)

Z<x<2Z Z<y<2Z
0<x<Z *0<y<Z*

The point *X* has coordinates (Z.Z) and is clearly black.



F«g«re 7.6 Definition of boundary inclusion depending on orientation. The heavy lines mark parts of the boundary whose points belong to the set.

Când discutăm despre o imagine în termeni de pixeli, obținem definiții consistente dacă folosim d-connectivity pentru pixeli albi și i-connectivity pentru pixeli negri. Această alegere a fost populară în literatură, dar eșuează atunci când avem de-a face cu imagini care conțin pixeli de mai mult de două culori. Este posibil să presupunem că dacă un set de celule conține anumite puncte ale limitei sale nu depinde de culoarea sa, ci
pe orientarea limitei. Pentru o unire de celule pătrate avem patru direcții ale normalei exterioare la graniță: 0, 2, 4. și 6 (în termenii notației din figura 7.4). Putem presupune că punctele limită aparțin mulțimii dacă și numai dacă una dintre următoarele două condiții este îndeplinită: (a) direcția normalei este 0 sau 2, (b) punctul este un colț cu

laturile în care normala este 0 și respectiv 2. Figura 7.6 ilustrează acest lucru. În această condiție, pixelii de aceeași culoare care se ating la un colț vor fi considerați a fi conectați dacă și numai dacă direcția bisectrizei este I. Descrierea analitică a unui dreptunghi sub această definiție va avea forma

$$a < x \wedge bc < y < d$$

Alegerea unei definiții în practică poate depinde de o serie de factori. De exemplu, în grafica raster, liniile sunt desenate cu pixeli care se ating doar la colțuri. Prin urmare, definiția compatibilă cu implementarea este de a presupune i-conectivitate pentru intensitatea primului plan și d-conectivitate pentru fundal. Acest lucru ar trebui să fie valabil chiar și în grafica color, deoarece primul plan și fundalul sunt parametrii de afișare bine definiți. Singura problemă potențială apare atunci când definiția fundalului și prim-planului este modificată în timpul formării afișajului, iar liniile de culori diferite se intersectează. De asemenea, este posibil să se utilizeze definiții mai complexe, dar acestea depășesc domeniul de aplicare al acestui text (vezi [3.PA], pp. 62-64).

7.4 O TEOREMĂ DE EȘANTIONARE PENTRU IMAGINI DE CLASA 2

Continuăm să studiem eșantionarea pozelor de clasa 2 cu grile pătrate. Dacă se cere ca fiecare regiune a unei anumite culori să fie suficient de mare pentru a conține o celulă de eșantionare la o orientare arbitrară, putem fi siguri că nicio regiune a imaginii originale / nu va lipsi din imaginea reconstruită *J,* dar pot exista totuși distorsiuni de formă considerabile. Definiția 7.4 impune o cerință mai strictă și vom demonstra că aceasta duce la păstrarea formei.

Definiția 7.4: O imagine de clasa 2 și o grilă de eșantionare pătrată ale cărei celule au dimensiunea laturii A sunt considerate *compatibile* dacă sunt îndeplinite următoarele două condiții: (a) Există un număr *d>V2h* astfel încât pentru fiecare punct de limită al fiecărei regiuni /? de o culoare dată, este posibil să găsim un cerc *C* cu diametrul *d* care este tangent la graniță și se află în întregime în /?. (b) Același lucru este valabil și pentru complementul lui *R.* □

Condiția de compatibilitate impune limite inferioare lățimii tuturor regiunilor și curburii contururilor acestora. În special, nu sunt permise colțuri. Aceasta poate părea o restricție serioasă, dar nu este

chiar asa. Într-adevăr, să presupunem că avem o clasă de obiecte ale căror conturi conțin colțuri. Putem alege o rază de curbură *r* și să înlocuim fiecare colț cu un arc de cerc cu raza *r.* Dacă r este suficient de mic, este posibil să nu fie percepută nicio distorsiune în forma obiectelor, în timp ce acum este posibil să găsiți o grilă compatibilă. Definiția 7.4 are avantajul că într-un număr de aplicații este posibil să se verifice dacă compatibilitatea este valabilă. De exemplu, fonturile de tipărire sunt adesea definite în așa fel încât să poată fi măsurate atât lățimea minimă, cât și curbura minimă.

Condiția de compatibilitate asigură anumite proprietăți care sunt date în continuare ca leme. Termenul de vecin, așa cum este folosit mai jos, este conform

Definiției 7.1, adică se aplică celulelor care se ating atât la colțuri, cât și la laturi.

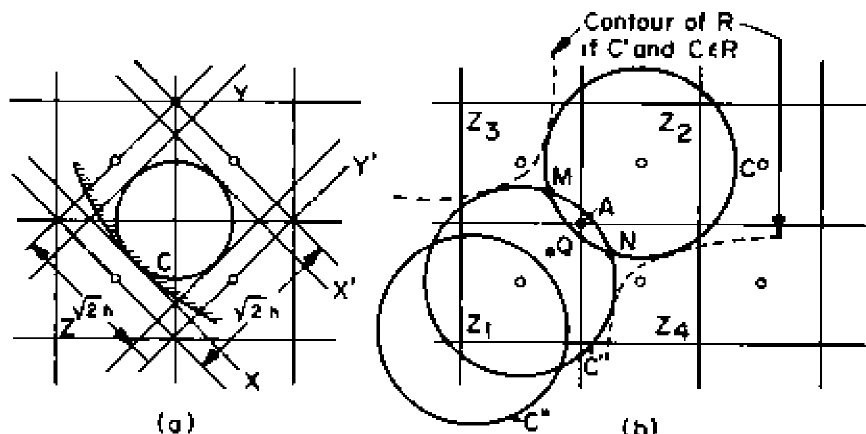


Figura 7.7 Ilustrații pentru demonstrațiile (a> Lema 7.1. și (b) Lema 7.2

Lema 7.1: În condiția compatibilității, dacă o celulă de eșantionare *Z* intersectează o regiune *R,* atunci fie centrul lui-Z este în *R,* fie există un vecin al lui Z al cărui centru este în /?.

Demonstrație: Fie *C* unul dintre cercurile cu diametrul din /?, care nu conține centrul lui *Z* , dar care se intersectează *cu Z.* Atunci *C* trebuie să fie tangent la o dreaptă X așa cum se arată în Figura 7.7a. Dacă *C* nu conține niciun alt centru, atunci trebuie să fie și tangent la liniile *X\ Y* și *Y'.* Dar distanța dintre fiecare pereche de linii paralele este mai mică decât *^h.* deci mai putin de *d.* O contradictie! □

Lema 7.2: În condiția compatibilității, dacă o regiune *R* conține centrele a două celule Z] și Z 2 care sunt vecine, atunci fie Z și Z 2 au o latură, fie există un vecin al lui Z și Z 2 cu centrul în *R* care împărtășește o latură cu fiecare dintre ele.

Dovada: Să presupunem că nu au o latură și că centrele lui Zj și Z 4 nu sunt în K. (Figura 7.7b.) Dacă colțul *A* ar fi în *R,* atunci ar trebui să fie în interiorul unui cerc cu diametrul *d* care este în întregime în *R. C* este un astfel de cerc care nu conține centrul lui Zj și Z 4 . Luați în considerare acum un cerc similar care conține centrul lui Z b Un astfel de cerc poate să intersecteze C (poziția C'), sau nu (poziția *C").* Dacă o face, atunci fie *M și / V* perechea punctelor de intersecție. Ambele aparțin lui *R* și orice cerc cu diametrul *d* care le conține trebuie să conțină și centrul fie al lui Z 3 , fie al lui Z 4 . care nu aparține lui *R* și aparține lui Z) așa cum se arată în figura 7.7. Atunci este imposibil să se găsească un cerc cu diametrul *d care să nu* conțină *Q și niciunul* dintre centrele lui Z și Z 2 , o încălcare a condiției de *compatibilitate*

Corolar: O consecință a condițiilor de compatibilitate este că toate regiunile conectate ale imaginii discrete vor fi conectate direct. □

Alungarea atingerii colțurilor nu este foarte surprinzătoare. A fost sursa majoră de incertitudine în deciderea conectivității, așa că a trebuit eliminată printr-un set de condiții care să asigure păstrarea conectivității , printre altele.

Propoziția 7.1: Fie 4 și *B* două puncte ale unei mulțimi *R* în imaginea analogică *I* și *K* o curbă care le unește și care se află în întregime în *R. R* este mapat într-o mulțime *R'* în imaginea reconstruită *J.* Atunci toate punctele lui *K* pot fi mapate pe o curbă *K'* care se află în întregime în *R'* și nici un punct corespunzător de *K* și *K'* mai departe de o distanță *d.*

Dovada: ipoteza de compatibilitate implică faptul că fiecare punct al lui *K* se află într-un cerc cu diametrul *d* conținut în întregime în *R.* Din cauza dimensiunii sale, un astfel de cerc trebuie să conțină centrul unei celule de eșantionare. Prin urmare, putem stabili o corespondență între toate punctele *K* și centrele celulare (Figura 7.8a). Datorită Lemei 7.2, putem găsi o cale de la centru la centru pentru toate aceste celule, iar această cale va traversa doar laturile și nu colțurile celulelor. Partea traseului din fiecare celulă poate fi pusă în corespondență cu partea de *K* care a fost mapată pe acea celulă (Figura 7.8b). În mod clar, putem stabili o mapare între

punctele lui *K* și traseul cu distanța punctelor respective care să nu depășească«/. □

Corolar: limitele regiunilor corespunzătoare din imagini / și *J* nu se află la o distanță mai mare de *d* unități atunci când imaginile sunt suprapuse. □

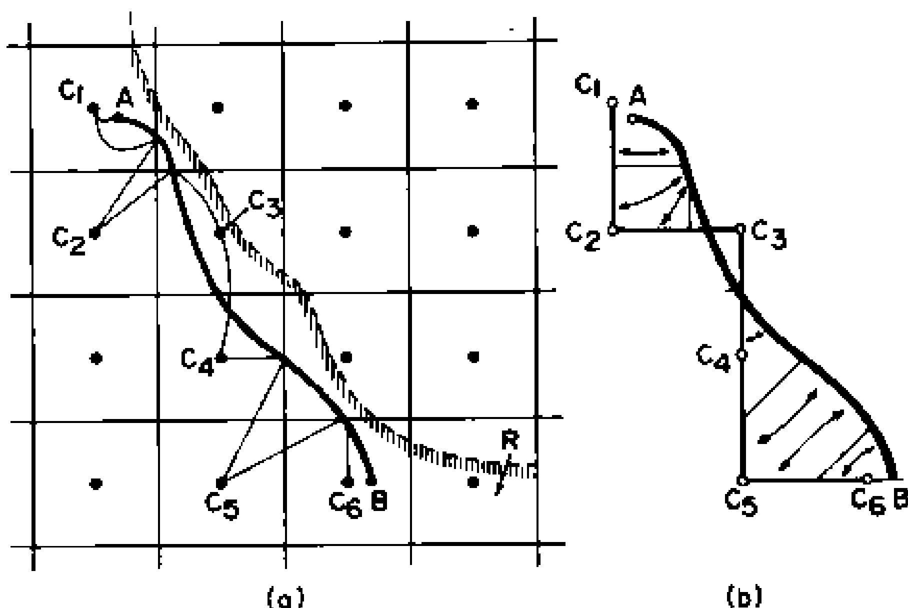


Figura 7.8 Ilustrații utilizate în demonstrarea Propoziției 7.1

Acest rezultat poate fi interpretat ca sugerând păstrarea formei între imaginea analogică originală și reconstrucția acesteia din mostrele sale. Cu toate acestea, cineva ar putea fi îngrijorat de posibilitatea ca unele regiuni să fi fost sparte sau fuzionate. Vom arăta în continuare că nu este cazul și că imaginile *I* și *J* sunt echivalente din punct de vedere topologic.

Teorema 7.1: Condiția de compatibilitate presupune păstrarea topologiei.

Dovada: Vom defini o mapare unu-la-unu între o mulțime și versiunea sa discretă care este continuă în ambele direcții. Dacă o celulă de eșantionare este conținută în întregime într-o regiune, atunci maparea necesară este identitatea. Dacă nu este, atunci luăm în considerare cazurile de excese și deficiențe. În exemplul din figura 7.9. regiunea triunghiulară *KLM* este o deficiență: o parte a celulei care nu aparține regiunii analogice. Regiunea *MAPENFQBM* este un exces deoarece face parte din regiunea analogică aflată în afara regiunii reconstruite. Dacă o celulă conține

sau este adiacent unei astfel de părți, este subdivizată în părți. Părțile celulei care nu aparțin nici uneia dintre aceste regiuni sunt considerate puncte ale regiunii analogice în care sunt mapate toate punctele celulei discrete. (Astfel, celula *DBFH* este redusă în *DCRGH,* partea de plafon *VLSMBD* în *VLTMCD* etc.) Părțile rămase sunt mapate în regiunile lor simetrice (în raport cu granița analogică), așa cum se arată în Figura 7.9.

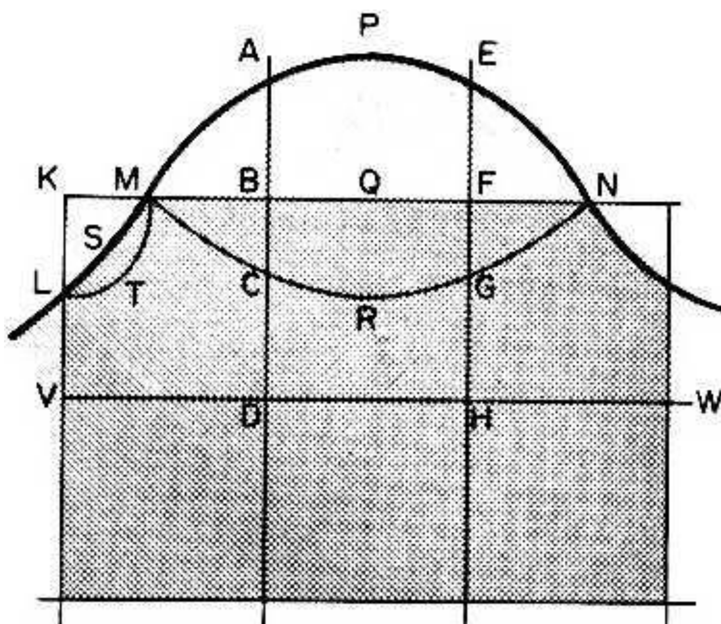


Figura 7.9 Ilustrație pentru demonstrarea Teoremei 7.1. Regiunea originală este umbrită la stânga (\), iar regiunea reconstruită la dreapta (/). Segmentele de linie *BD* **și** *FH ale* **imaginii discrete sunt mapate pe** *CD* **și** *GH* **ale analogului.** *AB* **și** *EF* **sunt mapate pe** *BC* **și** *FG, BQF* **pe** *CRG. APE* **pe** *BQF.* **etc. Punctele** *L. M* **și A' sunt fixe. Arcul** *LTM* **este mapat pe** *LSM,* **iar** *LSM* **pe arcul** *LKM (liniar pe bucăți).*

Datorită celor două leme putem fi siguri de continuitatea transformărilor. În special, Lema 7.1 garantează că orice exces va fi lângă o celulă cu aceeași culoare. Deoarece lema este valabilă atât pentru o mulțime, cât și pentru complementul său, ea asigură, de asemenea, că completarea deficiențelor nu va provoca fuzionarea niciunei regiuni. Lema 7.2 asigură că niciun colț nu se ating și, prin urmare, că toată transformarea poate fi făcută de-a lungul laturilor, așa cum se arată în Figura 7.9. □

Această teoremă, împreună cu Propoziția 7.1, implică atât faptul că

versiunea reconstruită a unei regiuni este echivalentă din punct de vedere topologic cu originalul analog și că fiecare curbă a unei regiuni poate fi mapată într-o curbă a celeilalte regiuni în așa fel încât distanța dintre punctele corespunzătoare să fie de o dimensiune comparabilă cu dimensiunea grilei de digitizare. Astfel, putem fi justificați să spunem că eșantionarea compatibilă păstrează forma.

Din punct de vedere topologic, condiția de compatibilitate implică faptul că toate seturile analogice sunt deschise și asigură că și seturile reconstruite sunt deschise. Această abordare pare să lase deschisă întrebarea cum să se ocupe liniile subțiri care au grosime zero teoretic și topologic nu sunt seturi deschise. Observăm că, în practică, liniile au grosime finită și nu există nicio modalitate de a le digitiza corect decât dacă îndeplinesc condiția de compatibilitate. Pe de altă parte, putem dori să facem o distincție între reconstrucțiile lor și reconstrucțiile mulțimilor care au o grosime finită din punct de vedere teoretic, precum și practic. Vom face asta în secțiunea 7.6.

7.5 TRASARE CONTURURI

Conceptul de graniță a unui plan stabilit în plan analog este bine definit. Este mulțimea tuturor punctelor care au proprietatea că, oricât de mică este considerată o vecinătate a acestora, conține puncte atât în interiorul, cât și în afara mulțimii. Conceptul corespunzător în planul discret nu este deloc clar. Vom da câteva definiții pentru a o clarifica. Rezervăm și termenul contur pentru planul discret, în timp ce termenul limită va fi folosit doar atunci când ne referim la mulțimi din planul continuu.

Definiția 7.5: *Conturul* sau *i-conturul* unui set conectat de pixeli *R* este definit ca mulțimea tuturor pixelilor din *R* care au cel puțin un d- vecin nu în *R.* d *-conturul* lui *R* este mulțimea tuturor pixelilor din *R* care au cel puțin un vecin care nu este în R. □

7.5.1 Trasarea unui singur contur

Pixelii unui contur pot fi traversați de o cale și este întotdeauna posibil să alegeți o cale închisă pentru parcurgere. În continuare, ori de câte ori ne referim la contururi, vom asuma o formă particulară de parcurgere, cea dată de Algoritmul 7.1. Acesta folosește conceptul de /V-neighbor (Definiția 7.1) cu notația prezentată în Figura 7.4. Se presupune că toate operațiile numerice pe *N* sunt modulo 8. Algoritmul poate fi descris în termenii unui observator care merge de-a lungul pixelilor aparținând mulțimii și selectează cel mai din dreapta pixel disponibil. Pixelul inițial *A* poate fi găsit în mai multe moduri, inclusiv de sus la

scanare de jos, de la stânga la dreapta a avionului. Trasarea se termină atunci când pixelul curent este același cu pixelul inițial. Deoarece aceasta este și condiția inițială, folosim *mai întâi steagul* pentru a distinge începutul de întoarcerea la punctul inițial. Bucla pașilor 5-9 este executată de cel mult trei ori pentru a evita rotirea în jurul unui set care are doar un pixel.

Algoritmul 7.1 Algoritmul de urmărire a conturului

Procedura *TRACER*

Notație: A **este punctul de pornire al conturului mulțimii** *R, C* **punctul curent a cărui vecinătate este examinată,** *S* **direcția de căutare conform codului din figura 7.4,** *mai întâi* **este flag care este adevărat numai când începe trasarea și** *găsit* **este flag care este adevărat când este găsit un următor punct de pe contur.**

1. Alegeți un punct /I din contur astfel încât vecinul său 4 să nu fie în set.
2. Setați punctul curent C la *A,* direcția de căutare *S* la 6 și steagul *mai întâi* la *adevărat.*
3. **În timp ce** *C* este diferit de *A* sau steagul este *mai întâi adevărat* , faceți pașii 3-10.

ÎNCEPE.

4. Setați *steagul găsit* la *fals.*
5. **În timp ce** *găsit* este *fals* **, faceți** pașii 5-9, de cel mult trei ori.

ÎNCEPE.

6. Dacă £, vecinul (.S'—1) al lui *C* este în *R,* atunci

ÎNCEPE.

7. Setați *C* la *B, S* la S—2 și *găsit* la *adevărat.*

Sfârşit.

8. **Altfel, dacă** *B* , vecinul S al lui *C,* este în *R* **, atunci** setați *C* la fiand *găsit* adevărat .
9. **Altfel, dacă** *B,* vecinul (S+l) al lui C, este în *R.* **atunci** setat *C* la *B* și s- a *găsit adevărat.*
10. În caz contrar, creșteți *S* cu 2.

Sfârşit.

11. Setați *mai întâi* la *fals.*

Sfârşit.

12. **Sfârșitul algoritmului.**

Algoritmul trebuie aplicat o dată pentru fiecare gaură a unei regiuni, în plus față de o aplicație pentru conturul extern. Prin urmare, trebuie combinat cu un algoritm de căutare pentru localizarea găurilor în interiorul regiunii. Vom prezenta un astfel de algoritm în scurt timp. Acest traver sal produce un i-path închis și urmează contururile externe în sens invers acelor de ceasornic și contururile găurilor în sensul acelor de ceasornic. Dacă se dorește o descriere a conturului ca rezultat al algoritmului, atunci se poate folosi căutarea

direcțiile plus coordonatele *xy* ale lui X. Aceste coordonate sunt scoase mai întâi și apoi, ori de câte ori un pixel este desemnat ca curent *(C).* iese valoarea 5. Astfel putem genera o codificare în lanț a conturului (vezi Secțiunea 1.2.3). Codificările lanțului pot fi, de asemenea, stocate intern și utilizate pentru traversarea interiorului regiunii în timp ce se caută găuri.

7.5.2 Trattrtal tuturor contururilor unei regiuni

Algoritmul 7.2 returnează toate contururile regiunii utilizând procedura TRACER *listată* mai sus. Algoritmul presupune că contururile găsite prin această procedură sunt plasate într-o coadă *Q.* Deoarece descrierile contururilor conțin atât perechi de coordonate xy, cât și coduri de lanț, trebuie avută o anumită atenție în implementarea lui 0. Pentru a simplifica descrierea algoritmului, presupunem că *Q* conține atât coordonatele *xy P* , cât și valoarea codului de lanț *c.* Dacă pixelul este un punct inițial al unui contur, atunci c este egal cu 8. În practică, s-ar putea avea o matrice care să stocheze numai coduri în lanț și să folosești simbolul 8 pentru a face referire la o matrice separată de puncte de pornire. Apoi coordonatele xy ale fiecărui punct nou ar putea fi obținute din punctul anterior plus informațiile codului de lanț . Algoritmul necesită, de asemenea, ca pixelii identificați ca aparținând conturului să fie marcați ca atare pe imagine. Pentru imaginile cu două niveluri presupunem că inițial pixelii regiunii sunt marcați cu I, iar cei care nu se află în regiune sunt marcați cu 0. Procedura *TRACER* poate fi modificată astfel încât atunci când un pixel este marcat ca punct curent, valoarea acestuia să fie incrementată cu 1. Apoi, la sfârșitul trasării, pixelii conturului vor avea valori 2 sau mai mari. În special, valoarea va fi unu plus numărul de ori pixelul a fost vizitat în timpul traversării. Aceste valori sunt folosite atunci când se caută găuri în interior. Pentru a evita căutarea părții din imagine în afara regiunii de interes, adoptăm următoarea politică.

După ce conturul extern a fost găsit și plasat în coada Q_t începem să examinăm conținutul acestuia din urmă. Dacă găsim un punct situat pe un arc descendent, începem o căutare spre dreapta. Astfel de pixeli pot fi caracterizați cu ușurință prin cerința ca anterior

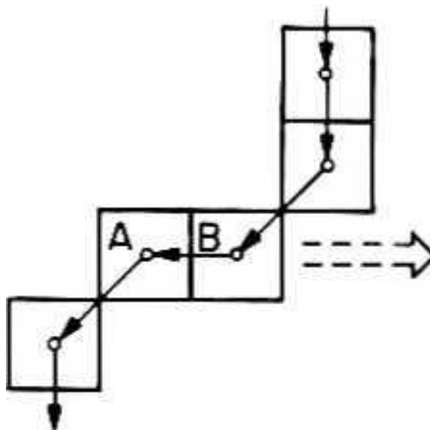


Figura 7.10 **Începutul unei scanări interne la un punct de inflexiune al conturului extern. Codul de lanț pentru porțiunea de contur afișată cu săgeți este 665456.** *B* **nu este selectat ca punct de plecare, dar** *A* **este.**

Elementul codului de lanț trebuie să aibă valori de la 4 la 7, în timp ce următorul element ar trebui să fie în intervalul de la 5 la 7. Includem valoarea 4 pentru primul element pentru a avea grijă de punctele de inflexiune, așa cum se arată în Figura 7.10. Deoarece direcția anterioară pentru primul element nu este cunoscută până când ultimul element al codului de lanț pentru acel contur nu este văzut, testul său ca punct de plecare ar trebui amânat până la sfârșit. (Nu includem această prevedere specială în lista algoritmului 7,2. pentru a simplifica descrierea acestuia.) În timp ce scanați de-a lungul direcției orizontale, trebuie să căutați fie începutul unei găuri, fie cealaltă parte a conturului exterior. Problema este complicată de faptul că pot exista pixeli care sunt comuni atât pentru conturul extern, cât și pentru conturul unei găuri, [dacă știam că acest lucru nu se poate întâmpla , atunci trebuie să căutați doar o secvență de doi pixeli cu valorile 01 ca început de găuri și o pereche cu valorile 2 (sau mai mari) și 0 pentru conturul extern. Posibilitatea de partajare a pixelilor necesită verificări mai complicate enumerate ca parte a algoritmului 7.2.

Algoritmul 7.2 Algoritmul complet de urmărire a conturului.

Notaţie:

Coada *Q* **conține atât adresele pixelilor** *P* **, cât și codurile de lanț** *c. ELIMITE* **ca în algoritmul 6.3.**

1. Găsiți conturul extern apelând **procedura** *TRACER.* Puneți punctele găsite și codurile de lanț într-o coadă *Q.*

2. **În timp ce** *Q* nu este gol, **faceți** pașii 3-8.

ÎNCEPE.

3. *(Pc} = REMOVE^}.* **Dacă** *c* este egal cu 8 *{P* este punctul de plecare I. atunci

setați c_0 egal cu *c* și eliminați încă un pixel: *(Pc) - REMOVE(C).*

4. **Dacă** c_0 este între 4 și 7 și *c* între 5 și 7, **atunci faceți** pașii 5-7.

ÎNCEPE.

5. Pornind de la *P,* căutați în direcția x și examinați triplete de pixeli succesivi. /I, *B, C.*

6. **Dacă** /I este 0. și *B* este I. **sau dacă** /I este 0.fi este 2. și *C* este 0. **apoi** utilizați £ ca punct de plecare și apelați **procedura** *TRACER* pentru a atașa conturul găurii la *Q.* După revenire, treceți la pasul 8.

7. **Altfel, dacă** ^ este 0, *B* mai mare decât 2. și *Q* este zero, sau **dacă** ^ este 1. B este 2 și *Q* este zero, **apoi treceți la** pasul 8.

Sfârşit.

8. Setați c_0 „ ^{c} -

Sfârşit.

9. **Sfârșitul algoritmului.**

Modelul 020 este folosit pentru a indica începutul unei găuri nescanate, deoarece un arc al conturului său poate coincide cu un arc al conturului extern (Figura 7.11a). Modelul 030 (sau 040. etc.) indică o gaură deja scanată de același tip și deoarece conturul exterior este îndeplinit. cautarea ar trebui sa se termine. Rețineți că regiunile cuprinse în întregime în găuri (Figura 7.1 lb) sunt ignorate. Motivul este că conturul găurii care conține o astfel de regiune va fi văzut mai întâi (linia de scanare *UU* în Figura 7.1 lb) și conturul acestuia trasat și marcat. Când o linie inferioară este scanată (de exemplu, kk în aceeași figură), scanarea se va termina din cauza apariției unui model 120 sau 030. Deoarece contururile găurilor, precum și contururile externe, opresc căutarea, contururile găurilor trebuie să fie și ele folosite ca pornire. Acest lucru se face automat, deoarece acestea sunt plasate și în coada *Q.*

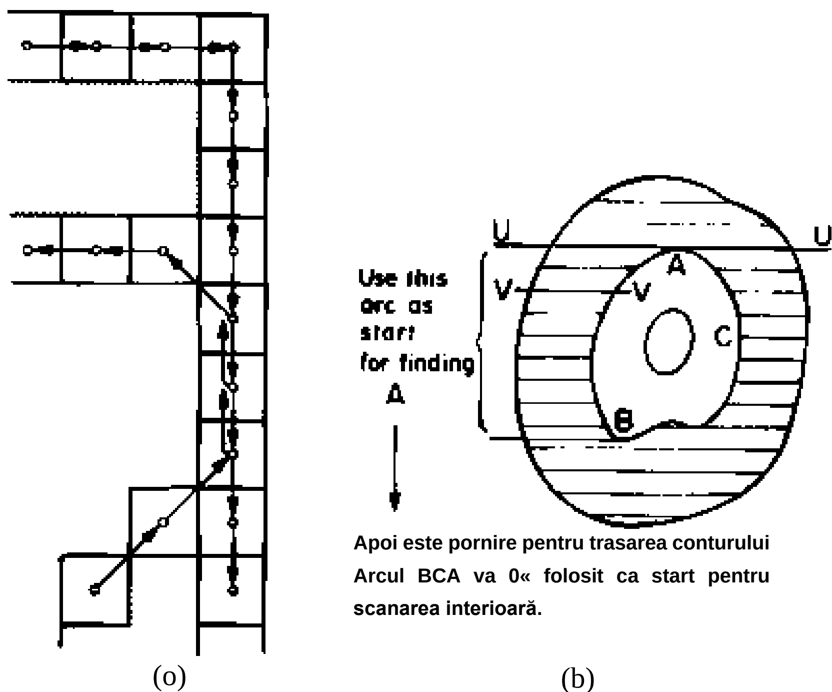


Figura 7.11 **(a) Un arc comun cu conturul exterior și conturul unei găuri; (b) Cum sunt scanate regiunile imbricate**

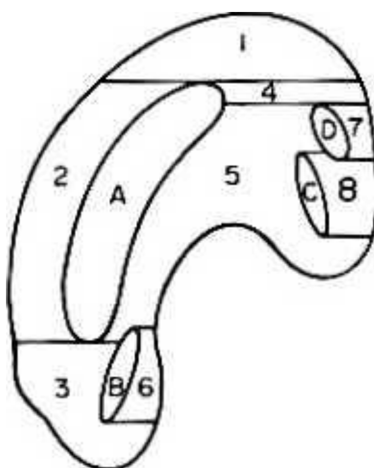


Figura 7.12 **Ordinea de scanare a interiorului unei regiuni**

Figura 7.12 arată ordinea în care este străbătut interiorul (regiunii. Regiunile imbricate pot fi găsite în scanările ulterioare ale imaginii. Pentru a crește eficiența unei astfel de căutări, coordonatele *xy* ale pixelilor la care s-a încheiat o scanare orizontală internă pot fi plasate într-o coadă care ar trebui căutată după găsirea contururilor fiecărei regiuni conectate.

Algoritmul 7.2 este eficient deoarece elementele imaginii care nu aparțin unui contur trebuie scanate o singură dată, iar cele aparținând unui contur sunt scanate doar de două ori. Nu este necesar să marcheze niciun pixel cu valori mai mari de 3, prin urmare poate fi modificat pentru a evita creșterea valorilor pixelilor dincolo de 3. Atunci este nevoie de doar doi biți per pixel pentru stocare.

7.6 CURBURI ȘI LINII PE O GRILĂ DISCRETE

Distincția geometrică intuitivă dintre curbe sau linii (care au o grosime zero) și regiuni plane (care au o anumită grosime finită) devine neclară pe o grilă discretă. Având în vedere un set de pixeli pe o astfel de grilă, nu este întotdeauna evident dacă setul a fost produs prin digitizarea unei curbe subțiri sau a unei regiuni întregi. În plus, definiția euclidiană a unei linii drepte ca distanță cea mai scurtă dintre două puncte eșuează dacă definim distanța în termeni de grilă. În exemplul din figura 7.13 prezentăm două căi i între punctele *A* și *B,* fiecare dintre ele constând din șase pixeli. Am pierdut unicitatea definiției euclidiene. Nu numai că, două astfel de linii se pot intersecta pe mai mult de un pixel. Cititorul poate verifica acest lucru luând în considerare intersecția uneia dintre liniile dintre >4 și *B* cu o linie similară între C și *D.* În funcție de alegerea anumitor linii, intersecția va conține oriunde de la zero la patru pixeli! Dificultatea nu dispare dacă decidem să găsim intersecția pe planul continuu și apoi să o mapăm pe discret. Liniile drepte analogice *AB* și *CD* dintre centrele pixelilor respectivi se intersectează într-un punct analog care ar putea fi mapat la oricare dintre cei patru pixeli marcați cu o etichetă îngroșată în Figura 7.13.

Problema acestor definiții nu este doar academică. În grafica raster, afișarea unei linii între două puncte este, de obicei, generată ca o cale i-conectată între ele, iar non-unicitatea poate fi o sursă de afișare slabă.

C							B
		2	2	2	2	3	
	3	1	1	1	1		
A							D

Figura 7.13 Ilustrarea neuniformității căii celei mai scurte dintre două puncte dintr-o grilă discretă. Pixelii unei căi sunt marcați cu I, iar pixelii celui de-al doilea cu 2'*. Pixelii marcați cu 3 aparțin ambelor căi.

Exemplul 7.1: Un sistem grafic raster afișează o linie **între** puncte făcând ca lungimile segmentelor de-a lungul direcției dominante să fie cât mai uniforme posibil. În Figura 7.14 pixelii liniei dintre punctele 0 și 7 sunt marcați cu o etichetă numerică. În mod clar, punctul / aparține acelei linii, dar linia dintre 0 și *f,* marcată cu litere mici, nu este o submulțime a liniei 07. în contrast cu definițiile geometriei euclidiene. (Pixelul d

este în afara liniei 07.) Să presupunem acum că 07 este o latură a poligonului afișată în culoare roșie. / este punctul de intersecție cu o dreaptă verticală *L.* și că vrem să colorăm toate segmentele de dreaptă în stânga lui *L* verde. Ceea ce vom vedea de fapt sunt niște pixeli verzi *(a* la/) precum și un pixel roșu (3) la stânga lui £. □

			d	4e	5f	6	7
0a	livre	2c	3				

Figura 7.14 **Pixelii liniei între 0 și 7 sunt marcați cu un număr, în timp ce pixelii liniei 0/** *{af}* **sunt marcați cu o literă mică. Pixelii cu două etichete aparțin ambelor linii.**

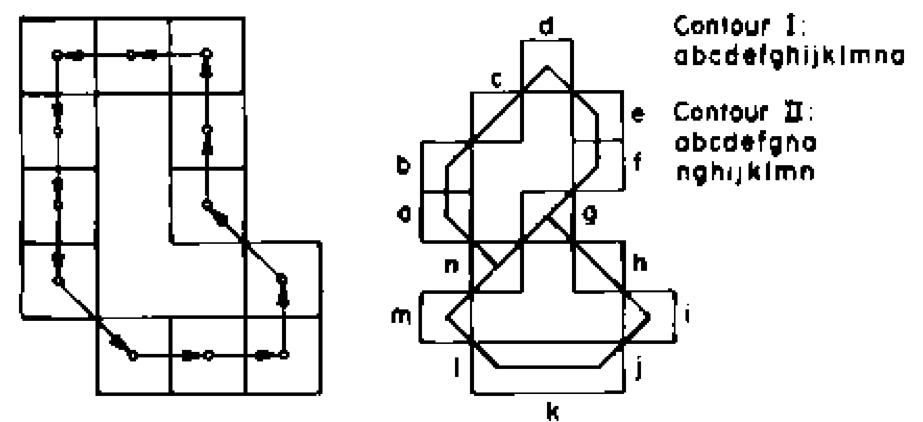
Definirea corectă a liniilor subțiri este, de asemenea, de interes pentru rezolvarea unor probleme precum umplerea contururilor (vezi capitolul 8) și subțierea regiunilor (vezi capitolul 9). Ambele probleme necesită execuția de operații geometrice pe o grilă discretă. Vom introduce un criteriu de distincție între curbele subțiri și regiuni groase. Apoi vom vorbi pe scurt despre modalitățile de a facilita rezolvarea problemelor geometrice.

7.6.1 Când un set de pixeli nu este o curbă

În loc să mergem direct la definiția regiunilor subțiri, vom defini mai întâi conceptul opus.

Definiția 7.6: Se spune că un set de pixeli *R* este *o regiune completă* dacă are mai mult de patru pixeli, conturul său i este o cale simplă, iar diferența dintre set și conturul său i este legată de d. □

Figura 7.15a prezintă un exemplu de regiune completă. În special, demonstrează că o regiune completă conține pixeli pe lângă cei ai conturului său . (Acesta nu ar fi fost cazul dacă nu am include ipoteza de aproximativ patru pixeli.) În termeni topologici, o regiune completă corespunde unui set deschis.



(a) (b)

Figura 7.15 Ilustrații ale definiției 7.6: (a) o regiune completă; (b) ambiguități în trasarea conturului

care ar apărea fără cerința de d-conectivitate.

În linii mari, o regiune este plină atunci când grila de eșantionare este suficient de fină încât nicio limită nu se dublează asupra ei înșiși. Cerința despre d-conectivitate este esențială dacă dorim să evităm situațiile ambigue precum cea prezentată în Figura 7.15b. Aceasta este o proprietate de dorit în situațiile în care am dori ca forma conturului să reflecte forma obiectului. De asemenea, este posibil să umpleți astfel de contururi printr-un algoritm simplu (secțiunea 8.3); prin urmare, proprietatea este importantă în grafica computerizată. Următoarea este o condiție suficientă pentru obținerea doar a regiunilor complete.

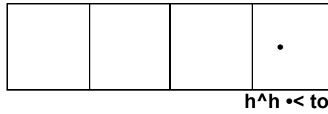
Teorema 7.2: Dacă cercul folosit în definiția compatibilității are diametrul VIOA în loc de doar V^A, atunci toate seturile de pixeli rezultate sunt pline.

Dovada: Figura 7.16 prezintă cea mai mică regiune care satisface condițiile teoremei: cuprinde în mod clar cel puțin nouă pixeli, care formează o regiune întreagă. Regiunile mai mari pot fi exprimate ca uniuni ale unor astfel de regiuni

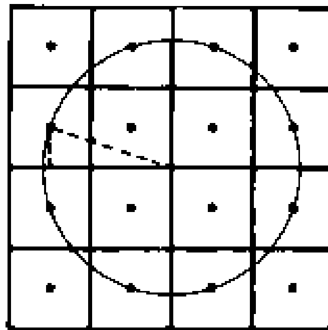
și, prin urmare, poate fi eșantionat ca seturi complete de pixeli. Dacă raza cercului este A?, aflăm că

* ˡ - (y>'+ (^-) ^{2} .sau. ^-|^.

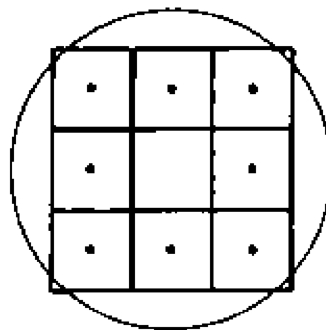
□



Regiunea cu lățimea <3h poate avea interiorul gol.



The smallest set with width /lOh nos a non-empty interior.



(b)

Aplicația practică a teoremei 7.2 este specificarea dimensiunii grilei, astfel încât digitizarea unui set dat de imagini cu două niveluri să aibă doar regiuni întregi. Fie *D* diametrul cercului mai mic înscris care îndeplinește condițiile din Definiția 7.4. Atunci *h*

A region With width /ion *6
has a non-empiy interior
regardless of grid location

trebuie să fie ales mai mic sau egal cu *D/V\Q.* Pe de altă parte, dacă ne-ar fi interesat doar conservarea formei, am fi putut alege *h* mai mic sau egal cu *D/\/2.* Raportul dintre aceste două dimensiuni de grilă este V5. Prin urmare, o digitizare în care toate regiunile

(c)

Figure 7.16 Illustration used in the proof of Theorem 7.2. (a) A region with width less than JA which has an empty interior, (b) Calculation for determining the minimum width, (c) A region with width greater than vTOA has a nonempty interior regardless of the grid location.

sunt pline va necesita de cinci ori mai multe puncte de eșantionare decât una în care nu toate regiunile sunt pline. Creșterea de cinci ori a cerințelor de stocare, precum și creșterea respectivă a timpului de procesare, este de obicei un preț prea mare de plătit pentru majoritatea aplicațiilor. Singura excepție majoră se găsește în grafică, unde rezoluția înaltă poate fi de dorit din motive estetice. Chiar și acolo, totuși, poate fi necesar să se ocupe de regiuni care nu sunt pline. Într-adevăr, vom folosi lipsa de plenitudine pentru a defini linii și curbe pe o grilă discretă.

7.6.2 Când un set de pixeli este o curbă

O caracterizare topologică riguroasă a curbelor subțiri este că acestea sunt mulțimi nevide care au un interior gol, adică toate punctele lor aparțin și ele graniței lor. Prin urmare definim:

Definiția 7.7: O *curbă* sau o *regiune liniară* pe o grilă discretă este un set de pixeli astfel încât toți aparțin și conturului mulțimii. □

Această definiție permite posibilitatea unor seturi de grosimea doi, precum cea prezentată în Figura 7.17. O definiție alternativă pentru seturile subțiri este de a cere să existe o cale unică între orice pereche de puncte ale acestora. Atunci exemplul din figura 7.17 nu va fi clasificat ca un set subțire. Se pare că în majoritatea aplicațiilor este mai logic să folosiți Definiția 7.7 decât alternativa. Vom discuta această întrebare mai detaliat în capitolul 9.

Figure 7.17 A set which is thin according to Definition 7.7.

7.7 MULTI PIXELI

Chiar dacă avem o definiție pentru seturile subțiri, nu suntem încă terminat cu problema. Într-adevăr, wc poate avea regiuni mixte care pot fi exprimate ca unirea regiunilor complete și a regiunilor liniare. Pentru aceștia, avem nevoie de o definiție care să se aplice la subseturile de regiuni și prin extensie la pixeli unici.

Am văzut că pixelii a căror valoare după trasarea unui contur este mai mare de doi joacă un rol special. Vom discuta proprietățile lor după introducerea anumitor definiții.

Definiția 7.8: *C-vecinii* unui pixel aparținând unui contur *C* sunt cei definiti ca elementele anterioare și următoare ale acestuia de-a lungul traversării definite de algoritmul 7.1. □

Rețineți că cei doi c-nighbors nu trebuie să fie distincti. Ele sunt întotdeauna distincte numai atunci când conturul este o cale simplă. Un exemplu al acestei definiții este prezentat în Figura 7.18.

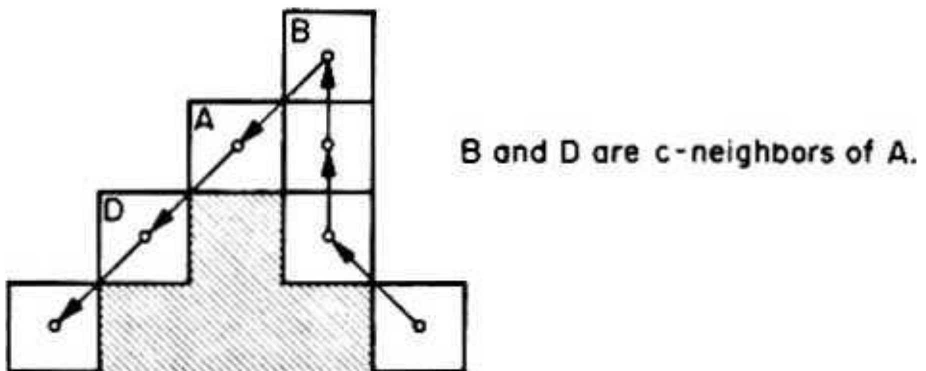


Figura 7.18 **Definirea c-vecinilor unui pixel de contur. B și D sunt c-vecini cu A. E nu este. E și A sunt vecinii c ai lui B.**

Definiția 7.9: Se spune că un pixel este *multiplu* dacă una sau mai multe dintre următoarele condiții sunt valabile:

(a) Este parcurs de mai multe ori în timpul trasării conturului.

(b) Nu are vecini în interiorul regiunii.

(c) Are cel puțin un d-vecin care aparține conturului, dar care nu este unul dintre c-vecinii săi. □

Figura 7.19 ilustrează motivația definiției. Pixeli multipli sunt pixeli în care se află două arce ale conturului sau unde se pliază un arc. Primele două condiții pot fi verificate cu ușurință folosind marcajele procedurii *TRACER* (Algoritmul 7.1). Într-adevăr, la

sfârșitul trasării pixelii simpli ai conturului vor avea valoarea exact 2. în timp ce mai mulți pixeli care satisfac condiția (a) vor avea valori mai mari decât 2. O a doua traversare poate fi utilizată pentru a verifica condiția (b) (niciun vecin cu valoarea 1) și condiția (c) (un d-vecin cu valoarea 2 sau mai mare și care nu este un c-vecin).

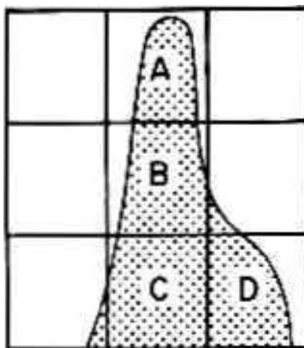


Figura 7.19 Ilustrare intuitivă a mai multor pixeli. A corespunde unei pliuri de limită (condiția b). B la două arce disjunse mapate pe același pixel (condiția a), iar C și D sunt pixeli adiacenți unde sunt mapate arce disjunse ale graniței (condiția c).

Chiar dacă detectarea mai multor pixeli prin traversări de contur este destul de simplă, poate să nu fie satisfăcătoare în aplicațiile în care se dorește procesarea paralelă. Deoarece verificarea condiției (c) depinde în mod esențial de o astfel de parcurgere secvențială avem o problemă cu Definiția 7.9. Intuitiv, ne așteptăm ca secvența să nu fie importantă deoarece verificăm pixelii care conțin două sau mai multe arce de contur sau unde un arc se îndoaie, configurații care sunt independente de ordine.

Continuăm cu o caracterizare a acestor pixeli care este independentă de ordine. Dacă un pixel este parcurs de mai multe ori în timpul trasării conturului unei mulțimi *R*, trebuie să fie doar pentru că nu există nicio modalitate de a trece de la o parte a *lui R*. R_t, la o altă parte, $R_{2\%}$ fără a trece prin acel pixel. Prin urmare, dacă pixelul este îndepărtat, gradul de conectivitate al setului *R* va fi redus cu cel puțin unul. În schimb, orice pixel a cărui îndepărtare va reduce conectivitatea lui *R* trebuie traversat de mai multe ori în timpul trasării conturului. Decizia dacă un pixel este esențial pentru conectivitatea setului poate fi luată prin examinarea celor opt vecini ai săi. Este ușor de arătat că *P* este esențial pentru conectivitate și numai atunci, dacă vecinătatea lui are una dintre formele prezentate în Figura 7.20 (sau oricare obținută prin rotație de 90'). Acolo deci. un pixel poate fi clasificat ca traversat prin multiplicare dacă și numai dacă este

cartierul se conformează unuia dintre aceste modele. În acea diagramă (și în cele

ulterioare) se utilizează următoarea notație pentru valorile pixelilor:

Definiția 7.10: Un număr denotă valoarea pe care ar fi dat un pixel în timpul traversării conturului: 1 pentru pixelii interiori. 2 pentru pixelii de contur parcurși o singură dată, 3 sau mai mult pentru pixelii de contur parcurși de mai multe ori. Un număr urmat de semnul plus (+) indică un pixel cu o valoare cel puțin egală cu acel număr. 0 este pentru pixelii care nu sunt în set. O literă, cu excepția lui *X,* indică faptul că pixelul poate avea orice valoare diferită de zero. Eticheta *X* reprezintă orice valoare (condiția nu-ți pasă). Un grup de pixeli marcați cu același simbol (de ex. A) are proprietatea că cel puțin unul dintre pixeli are o valoare mai mare decât zero. Simbolul sur (•) denotă un pixel care poate avea orice valoare diferită de 1. □

Astfel, în Figura 7.20 (modelul din stânga), cel puțin unul dintre cei trei pixeli marcați cu A este diferit de zero și cel puțin unul dintre cei trei pixeli marcați cu B este diferit de zero. Rețineți că putem decide dacă un pixel aparține conturului fără traversare. Trebuie doar să aibă un vecin d care este zero.

	A	A	A			A	A	A	
	0	P	0			A	P	0	
	B	B	B			A	0	2	

Figura 7.20 Configurații de vecinătate ale pixelilor care sunt esențiale pentru conectivitate. Vezi textul pentru o explicație a notației.

Astfel am demonstrat:

Propunerea 7.2: Condiția (a) din Definiția 7.9 poate fi înlocuită cu cerința ca vecinătatea de opt puncte a unui pixel să prezinte cel puțin unul dintre modelele din Figura 7.20 (sau cele obținute din acestea prin rute de 90'). □

Condiția (b) poate fi verificată cu ușurință folosind un algoritm paralel. Niciunul dintre cei opt vecini nu poate avea eticheta I. Acest lucru va păstra tonuri unice (adică pixeli fără vecini) și punctele finale (pixeli cu exact un vecin). Ultima observație sugerează că verificarea celui de-al doilea model din figura 7.20 poate fi simplificată permițând tuturor pixelilor marcați A să fie zero. Rețineți că condiția (b) salvează, de asemenea, toate liniile de lățime doi, cum ar fi cea prezentată în Figura 7.17,

Ne rămâne cu condiția (c). Dacă un pixel îndeplinește această condiție ca

precum și (a) sau (b), atunci va fi clasificat ca multiplu oricum. Astfel, putem căuta un criteriu care este mai restrâns decât condiția (c). atâta timp cât pixelii pe care îi ratează sunt prinși de celelalte două condiții. Vom arăta că pentru astfel de pixeli condiția poate fi înlocuită cu o verificare pentru un model de pixeli 3X3. În acest scop, începem cu un model mai mare (4X4) și luăm în considerare toate configurațiile posibile din acesta care vor da naștere la condiția (c). Configurația generală este prezentată în Figura 7.21.

	un Q		
«1	a_2	° 1	
q	q	q	

Figura 7.21 Etichetarea pixelilor utilizată în **dezvoltarea** unei forme alternative de Definiție 7.9

Fără pierderea generalității presupunem că c_2 este pixelul curent cu valoarea 2 și că C| este vecinul său d cu valoarea 0. (Reamintim că toți pixelii contour au un astfel de vecin.) Alte configurații pot fi obținute prin rute de 90'. Deoarece pixelul c_2 satisface condiția (c) din Definiția 7.9, trebuie să aibă și un vecin d cu valoare mai mare sau egală cu 2.

Propoziția 7.3: Orice pixel care îndeplinește condiția (c) din Definiția 7.9 dar nu (a) sau (b) face parte dintr-un model 022+ de-a lungul unei linii verticale sau orizontale.

Dovada: Vom demonstra proprietatea pixelului c_2. În primul rând wc luăm în considerare pixelul cj și wc arată că, dacă nu are valoarea 2+ (astfel încât poziția pro să nu fie satisfăcută de-a lungul unei linii orizontale), atunci va exista o linie verticală prin c_2 cu acea proprietate. Dacă c_3 are valoarea zero, atunci pixelul c_2 este fie traversat prin multiplicare, fie un punct final și, prin urmare, este clasificat ca multiplu în condițiile (a) sau (b). Să presupunem că are valoarea 1. Aceasta înseamnă că pixelii UJ, **q** și à nu pot fi zero, altfel q ar fi în contur. Din cauza presupunerii noastre că condiția (c) este valabilă, fie a_2, *fie* b_2 trebuie să aibă valoarea 2 sau mai mare. Trebuie să luăm în considerare doar primul dintre acești pixeli din cauza simetriei. Fie ca un $_2$ să aibă valoarea 2+. Dacă a, are valoarea zero, atunci a_2 va fi un c-vecin al lui c_2 și asta înseamnă că 6_2 trebuie să fie, de asemenea, cel puțin 2 și b_x non-zero, configurația simetrică. Astfel, continuăm presupunând că $a_($ este diferit de zero. Acum a_2 este un pixel de contur care are d-vecini c_2, a_b și Oj care nu sunt zero. Prin urmare, pixelul a_0 trebuie să fie zero și avem modelul 022+ în pixeli *a^* a_2 *și* c_2. □

Datorită acestui rezultat putem presupune, fără pierderi de generalitate,

că c_3 din figura 7.21 are valoare mai mare sau egală cu 2. Examinăm în continuare configurațiile posibile ale pixelilor marcați a_b a_2, și o_3 și a celor marcați 6_b h_2, și 6,.

Cazul *i:* Atât a_3 cât și *by* sunt diferite de zero. Atunci c_4 trebuie să fie zero, altfel c_3 nu ar putea fi un pixel de limită. Dacă toți pixelii d_b d_2, $Z>_b$ și b_2 ar fi zero, atunci a_3 și *by* ar avea valoarea 2+ și c_2 ar fi fost clasificați ca multiplu în condiția (b) (niciun vecin etichetat I). Prin urmare, trebuie să avem modelul din figura 7.22a cu cel puțin unul dintre acești patru pixeli diferit de zero. Numărul de configurații posibile pentru acest model este destul de mic și se poate verifica că în toate) condiția (c) este îndeplinită. (Un exemplu este prezentat în Figura 7.22b.)

A	A	P	X		0	0	2	X
0	2	2+	0		0	2	2	0
A	A	Q	X		2	1	1	2

(a) (b)

Figura 7.22 **Ilustrarea cazului i: modelul general în stânga și un caz special în dreapta**

Cazul u. Atât oj cât și *by* sunt zero. Atunci este ușor de verificat că c_2 nu poate avea vecini etichetați I. și, prin urmare, este tratat în condiția (b). Figura 7.23 prezintă modelul pentru acest caz.

•	•	0	X
0	2	2 +	X
•	•	0	X

Figura 7.23 **Ilustrarea cazului ii. Niciunul dintre pixelii marcați cu 2 și 2+ nu este esențial pentru conectivitate, dar dacă ambii sunt îndepărtați, conectivitatea este schimbată.**

Cazul iii: a $_}$ nu este zero și ^j este zero. (Cazul dj — 0 și bj > 0 este similar și nu îl vom discuta.) Dacă atât 6, cât și b_2 sunt zero, atunci conturul trece de la c_2 la c_b și c_2 nu poate fi un pixel multiplu decât dacă îndeplinește condiția (c). Modelul este prezentat în Figura 7.24a cu un caz special în Figura 7.24b. Dacă *b* și b_2 nu sunt ambele zero, dar d] și d_2 sunt, atunci c_2 nu va avea un vecin interior și este tratat în condiția (b).

X	X	p	X			2	2	1	X
0	2	2 +	X			0	2	2	1
0	0	0	X			0	0	<u>0</u>	<u>2</u>

(a) (b)

Figura 7.24 Ilustrarea cazului iii: Modelul general în stânga și un caz special în dreapta

Această discuție epuizează toate configurațiile posibile care apar în dreptunghi de 3X4 pixeli. Dacă condiția (c) va fi îndeplinită, trebuie îndeplinită într-o astfel de zonă. Prin urmare, am dovedit:

Propoziţia 7.4: Condiţia (c) este echivalentă cu căutarea unui model de forma dată în Figura 7.25 (plus cele obţinute prin rotaţii de 90').

O	O	C
0	2	2 +
B	B	C

Figura 7.25 Model echivalent cu condiția (c). Cel puțin unul dintre pixelii marcați C trebuie să fie diferit de zero. Dacă ambii pixeli etichetați C sunt diferit de zero, atunci valoarea pixelilor etichetați A și B poate fi orice . În caz contrar, cel puțin unul dintre membrii fiecărei perechi marcați A sau B trebuie să fie diferit de zero.

Figura 7.26 prezintă câteva modele specifice de pixeli. Pixelul curent este marcat cu un simbol aldin dacă este multiplu.

0	2	1	1	eu	1			0	2	eu	1	1	1
0	0	2	1	1	1			0	0	2	1		1
0	0	0	P	2	1			0	0	0	**P**	2	1
0	0	0	2	0	2			0	0	0	0	0	2
0	0	0	0	2	1			0	0	•o	0	2	eu

Figura 7.26 Stânga: exemplu de pixeli multipli (P) care satisface condiția (c). Rețineți că pixelul P nu satisface nici una dintre celelalte două condiții . Dreapta: un caz în care această condiție nu este îndeplinită pentru pixelul P.

Astfel, am găsit o altă caracterizare a mai multor pixeli.

Teorema 7.3: Un pixel este multiplu dacă îndeplinește cel puțin una dintre următoarele trei condiții.

(a) Vecinătatea sa se conformează cu oricare dintre modelele din Figura 7.20 (sau cu cele obținute prin rotații de 90*). Pixelii etichetați cu A în al doilea model pot

avea orice valoare.

(b) Are cel mult un vecin diferit de zero sau nu are vecini etichetați 1.

(c) Vecinătatea sa satisface modelul din Figura 7.25 (sau cele obținute prin rute de 90'). □

În mod clar, este posibil să se verifice aceste condiții fie în mod paralel, fie secvenţial. Prezentăm un rezultat care leagă conceptul de pixeli multipli de linii subțiri sau curbe.

Propunerea 7.5: O regiune liniară este formată în întregime din mai mulți pixeli. În schimb, dacă o regiune constă numai din mai mulți pixeli, aceasta este liniară.

Dovada: Deoarece setul nu are alți pixeli decât cei ai conturului său, niciunul dintre pixeli de contur nu va avea vecini în interiorul regiunii și, prin urmare, condiția (b) din Definiția 7.9 este valabilă. Dovada inversului este trivială, deoarece un pixel multiplu este întotdeauna un pixel de contur. □

Analiza de mai sus este utilă pentru detectarea liniilor sau operarea pe ele pe o rețea discretă. Nu am abordat problema generării unor astfel de afișaje din ecuații de linii sau curbe. Acest lucru va fi tratat în capitolul 10.

7.8 O INTRODUCERE ÎN ANALIZA FORMELOR

Analiza formei este una dintre problemele fundamentale în recunoașterea modelelor și este, de asemenea, de interes pentru grafica interactivă. Este relevant atunci când cineva trebuie să ia o decizie pe baza formei obiectelor pe care le vede. Definiția psihofizică precisă a ceea ce este forma este în afara domeniului de aplicare al acestui text. Cu siguranță, termeni precum „alungit” sau „colț ascuțit” se referă la formă. S-ar putea defini forma implicit afirmând că este informația conținută într-o imagine cu două niveluri, fără a lua în considerare culoarea regiunilor. Această definiție se limitează la forma siluetelor, dar este adecvată pentru multe aplicații și în special pentru recunoașterea caracterelor alfanumerice.

Se pot distinge două moduri de recunoaștere a formei. Într-una, o persoană se uită la obiectul total și ia o decizie pe baza structurii generale. Acesta este în mod obișnuit cazul în recunoașterea literelor imprimate manual, în special chineze, unde se identifică linii sau alte elemente de bază. Într-un alt mod se examinează conturul siluetei, de obicei căutând colțuri, proeminențe, intruziuni,

și alte puncte de curbură mare. Un exemplu este recunoașterea siluetelor profilurilor umane sau verificarea defectelor contururilor elementelor de circuit pe plăcile de cablaj imprimate.! Desigur, există multe situații în care trebuie utilizate ambele moduri: un desen ingineresc conține linii și litere recunoscute după structura lor, precum și cercuri sau hexagoane care trebuie să fie distinse unele de altele pe baza contururilor lor. Majoritatea metodologiilor din trecut au fost orientate spre unul dintre cele două moduri și, din moment ce teoretic este posibil să le aplici oricărui obiect, acestea au fost folosite în cazurile în care nu erau potrivite. În mod ideal, s-ar dori să existe o metodă mixtă care să se adapteze la modul cel mai potrivit pentru obiectul examinat.

Ne vom limita aici la o discuție despre metodele simple de analiză a conturului,

așa cum sunt utilizate pentru descrierea formei obiectelor găsite în imaginile de clasa 2. Metodele structurale vor fi revizuite pe scurt în Capitolul 9. Curbatura este o caracteristică importantă în analiza formei, nu numai atunci când este utilizată direct, ci și atunci când este utilizată indirect. Din păcate, o măsurare directă a curburii nu este întotdeauna fezabilă din cauza zgomotului. Formula dată în majoritatea textelor de calcul necesită luarea unei derivate a doua și, prin urmare, nu poate fi folosită în nicio situație practică. Pe de altă parte, este posibil să se estimeze raza de curbură prin construcția geometrică prezentată în Figura 7.27. Într-adevăr, dacă^.fl. și *C* sunt puncte pe o curbă. *M* și *N* sunt punctele mijlocii ale intervalelor *AB* și respectiv *BC* , iar *K* punctul în care se intersectează normalele de la A/ și *N* , apoi *R.* lungimea lui *BK,* este egală cu raza unui cerc care trece prin *A. B* și C.

Dacă unghiul *ABC* este egal cu 0, 0 indică complementul său, iar lungimile lui *AB* și *BC* sunt fiecare egale cu 2e, atunci un calcul simplu arată că

™s(y) sin(-^)

(vezi problema 7.7). Din figura 7.27 și din ecuația (7.1) reiese clar că *R* este o funcție crescătoare a lui 0. Dacă 0 este mic, așa cum este adesea cazul, atunci curbura *c* care este definită ca inversul razei de curbură va fi dată aproximativ de

$$c = \frac{\psi}{2e} = \frac{\pi - \phi}{2e}$$

t Aceasta nu trebuie confundată cu așa-numita percepție a formei *Ge^tah în care,* potrivit psihologilor, se recunoaște forma unei figuri fără a privi părțile.

11 este posibil să se elimine unele dintre efectele zgomotului prin calcularea curburii nu pe pixeli succesivi, ci pe tripleți, unde distanța e este mare în comparație cu frecvența zgomotului spațial. Există multe variante ale acestei tehnici, dar nu ne vom ocupa de ele aici. O determinare indirectă a maximelor de curbură poate fi realizată prin - aproximări poligonale, deoarece colțurile vor avea tendința de a fi plasate lângă astfel de maxime.

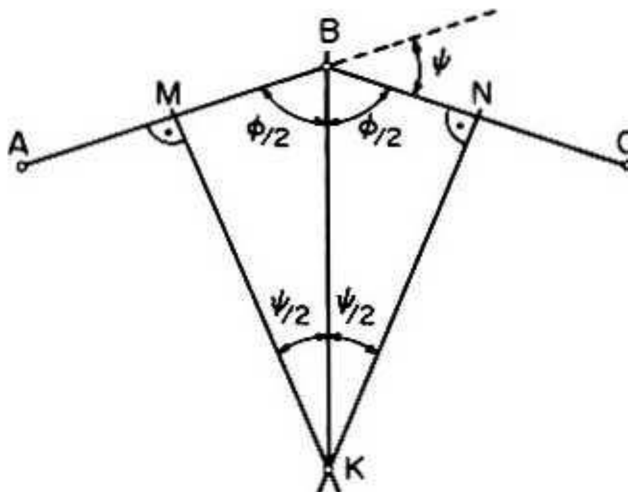


Figura 7.27 Construcție utilizată pentru estimarea razei de curbură

Este posibil să procedați cu o analiză a conturului în mai multe moduri. La cel mai de jos nivel, metodologia se bazează pe o reprezentare primitivă, cum ar fi un cod în lanț. La un nivel superior reprezentarea constă într-o aproximare prin bucăți de curbe netede (ex. prin B-splincs). Acesta din urmă este de preferat atunci când datele sunt zgomotoase și când se caută caracteristici care implică o mare parte a conturului. Prima este cea mai bună pentru date cu niveluri scăzute de zgomot și caracteristici localizate. Aproximațiile poligonale au fost folosite adesea nu numai pentru că detectează maximele de curbură, ci și pentru că sunt mai simplu de implementat decât alte tehnici de ajustare a curbei. Vom amâna discutarea unor astfel de tehnici până după capitolele 11 și 12, care discută spline și aproximări.

Vom discuta despre detectarea caracteristicilor locale folosind codul de lanț diferențial din Secțiunea 1.2.3, deoarece oferă expresii care sunt independente de orientare. Dacă conturul este suficient de neted, singurele simboluri prezente vor fi 0 și ±1. Prezența lui +2 implică un unghi de 90*, în timp ce cea a lui ±3 implică un unghi de 45*. Ar fi

simplu de asociat apariția unor astfel de simboluri cu curbura max ima, dar acest lucru nu este suficient pentru a localiza toate astfel de maxime. De exemplu, secvența 01110 reprezintă, de asemenea, un unghi de 45'.

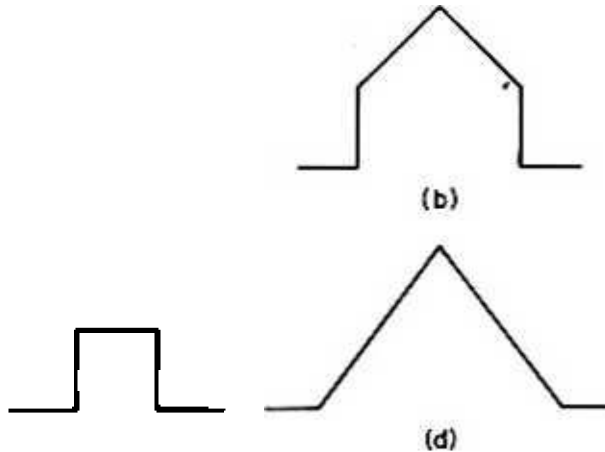


Figure 7.28 Examples of notches. The respective differential chain codes are as follows:

(a) • • • 0 + 1-2+10 • ,

(b) ■ ■ • 0+2-1-2-1+20 • .

(c) • ■ • 0+2-2-2+20 • ■ .

(d> • • • 0 + 10-20 + 10 • ■ .

Există patru clase de arce de interes: linii drepte, colțuri, arce aproximativ circulare și crestături. O linie dreaptă de-a lungul uneia dintre direcțiile codului de lanț are forma 0". Dacă nu este în una dintre aceste direcții, atunci va fi de forma (0"+l—1)* sau {O'" —1 + 1)*.t Din cauza zgomotului, codul real poate să nu fie la fel de regulat, dar va fi întotdeauna caracterizat prin apariția +1 — 1 sau —1 +), arcul circular va fi pe de altă parte a perechilor. (0"+i)* sau (0"—1)*, principala caracteristică este că ± T-urile vor apărea individual și pentru o anumită curbă, semnul lor va fi fix. Un colț va conține fie una dintre valorile mari ale codului, fie o secvență de l cu același semn un defect. În ceea ce privește codul de lanț, o crestătură este o secvență de valori mari ale codului care se adună până la 0 (sau cel mult ± 1).

t De fapt, m poate varia cu unul de la segment la segment. Vezi |7.WU| pentru o caracterizare de precité a descrierilor de linii drepte prin coduri de lanț.

Se pot veni cu algoritmi simpli pentru detectarea unor astfel de caracteristici, iar aplicabilitatea lor depinde de cât de lipsite de zgomot sunt datele. O tehnică utilă se bazează pe definirea expresiilor regulate sau (echivalent) automatelor finite care se potrivesc cu astfel de caracteristici. De exemplu, următoarea expresie va recunoaște unele dintre crestăturile prezentate în Figura 7.28:

0(+1 sau +2)(un șir de cel mult două simboluri negative)(+1 sau +2)0

Recunoașterea dacă o secvență lungă reprezintă o linie dreaptă sau un arc

circular este o problemă mai dificilă. Pentru astfel de caracteristici la scară mare, cel mai bine este să utilizați alte tehnici, cum ar fi potrivirea curbei (vezi Capitolul 12).

7.9 NOTE BIBLIOGRAFICE

Primele discuții sistematice ale problemelor implicate în definirea conectivității și a altor concepte topologice pe o grilă discretă pot fi găsite în [7.RO] și [7.MY). Trasarea conturului este o procedură destul de simplă, cu excepția problemelor asociate cu conectivitatea discretă . Vezi (7.SO) pentru o discuție. |7.MR] descrie un algoritm care nu necesită etichetarea imaginii. [7.PA1) prezintă un algoritm care face trasarea pe baza unei codificări a lungimii de rulare a imaginii.

Tratarea detaliată a subiectului analizei formei depășește sfera acestui text. Cititorul este referit la literatura de recunoaștere a modelelor, unde subiectul este tratat mai mult. (7.FUI), I7.FU2), (2.HA) și (3.PA| sunt texte avansate, în timp ce [7.PA2] și |7.PA3) oferă recenzii ale literaturii de specialitate. Cei interesați de aplicații specifice ar trebui să consulte literatura despre aplicații. De exemplu, o trecere în revistă a lucrărilor timpurii privind recunoașterea caracterelor poate fi găsită în (7.UL). Exemple de analiză a contururilor pot fi găsite în multe lucrări. Utilizarea concavităților pentru descrierea simbolurilor scrise este subliniată în [7.AP] și)7.YM] printre altele.

7.10 LITERATURA RELEVANTA

|7.AP) Ali. F. și Pavlidis, T. „Recunoașterea sintactică a numerelor scrise de mână”, *IEEE Trans. Sisteme. Om. Cibernetică.* **SMC-7 (1977), pp. 537-541.**

(7.FU11 Fu, KS *Syntactic Methods in Pattern Recognition.* **New York: Academic Press, 1974, 295 pp.**

I7.FU2| Fu. KS (ed.) *Syntactic Pattern Recognition. Aplicații.* **Heidel berg: Springer-Verlag. 1977, 270 pp.**

(7.KE) Kelley. JL *Topologie generală.* **Princeton, NJ.: Van Nostrand. 1955.**

[7.MRJ Morrin, TH „Chain-Link Compression of Arbitrary Black-White (mages”, *CGIP.* **5 (1976), pp. 172-189,**

[7.MY| Mylopoulos, J. și Pavlidis, T. „Despre proprietățile topologice ale spațiilor cuantizate”. *JACM.* **1« (aprilie 1971), Partea I pp. 2 39-246, Partea II pp. 247-254.**

17.PA1] Pavlidis, T „Algoritmul de urmărire a limitelor de stocare minimă și aplicarea sa în inspecția automată”, *IEEE Trans. Sisteme. Man, Cybernetics,* **SMC-8 (1978), pp. 66-69.**

[7.PA2] Pavlidis. T. „O revizuire a algoritmilor pentru analiza formei”, *CGIP.* **7 (aprilie 1978), p. 243-258.**

(7.PA31 Pavlidis, T. ”Algorithms for Shape Analysis of Contours and Waveforms.” *IEEE Trans. Pattern Analysis and Machine Intelligence,* **PAM1-2 (iulie 1980), pp. 301-312.**

17.ROJ Rosenfeld. A. „Conectivitate în imagini digitale”, *JACM,* **17 (1970) pp. 146-160.**

[7.SO1 Sobel, I. „Neighborhood Coding of Binary Images for Fast Contour Following and General

Binary Array Processing", *CGIP,* **8** (august 1978), pp. 127-135.

(7.UL1 Ullmann. JR „Picture Analysis in Character Recognition." în *Digital Picture Analysis.* (A. Rosenfeld, ed.). Heidelberg: Springer 1976, pp. 295-343.

[7.WU] Wu, LD „Despre conjectura lui Freeman despre codul în lanț al unei linii", *Proc. Al cincilea intern. Conf, pe Pattern Recognition,* Miami Beach. decembrie 1980, p. 32-34. (Publicat de IEEE Computer Society. Catalog IEEE Nr. 80CHI499-3.)

[7.YM1 Yamamoto. K. și Mori, S. „Recunoașterea caracterelor imprimate manual prin metoda punctului exterior", *Proc. Al patrulea intern. Joint Conf, privind recunoașterea modelelor.* Kyoto, noiembrie 1978, p. 794-796.

7.11 PROBLEME

7.1. Găsiți transformata Fourier a unei forme de undă pătrate care are lățimea minimă *H.* adică fiecare interval în care valoarea este constantă are cel puțin *H* unități lungi. Repetați, presupunând că modificările valorii apar numai la multipli întregi ai unei cantități *h.* Comparați cele două transformări și studiați modul în care dimensiunea raportului *H/h* afectează similitudinea lor.

7.2. Încercați să estimați intervalul de eșantionare adecvat pentru digitizarea textului acestei pagini.

7.3. Creați un algoritm pentru numărarea numărului de blob-uri (seturi conectate de o anumită culoare) pe o imagine de clasa 2.

7.4. O problemă practică comună este necesitatea de a mări sau reduce a imagine pe două niveluri. De exemplu, cineva poate avea un set de matrici pe două niveluri care descriu caractere alfanumerice și doriți să îl afișați la două ori sau jumătate din dimensiune. Puteți concepe un algoritm pentru a face astfel de transformări păstrând în același timp forma? În special , având în vedere un set, calculați care ar fi cea mai mare reducere posibilă.

7.5. Implementați algoritmul 7.2 și utilizați-l pentru a număra numărul de blob-uri care sunt seturi complete, seturi liniare sau seturi mixte.

7.6. Fie *R* un set conectat de pixeli și fie *C(R)* pixelii conturului său. Arătați că dacă eliminați toți pixelii lui C(/?) cu excepția celor care sunt multipli, pixelii rămași formează un set care nu este doar conectat, ci și echivalent topologic cu /?. Ce poți spune despre asemănarea formei?

7.7. Demonstrați ecuația (7.1).

7.8. Introduceți o măsură a netezirii conturului folosind codul de lanț diferențial (de exemplu, proporția valorilor diferite de zero ca procent din total). Apoi încercați să rezolvați problema 7.4, astfel încât valoarea acestei măsuri să rămână constantă sau aproape constantă.

Capitolul 8

UMPLUREA CONTURULUI

8.1 INTRODUCERE

Una dintre cele mai frecvente probleme în grafică și analiza imaginilor este găsirea interiorului unei regiuni atunci când este dat conturul acesteia, ic, transformarea unei imagini de clasa 3 într-o imagine de clasa 2. De exemplu, orice algoritm de umbrire presupune soluția acestei probleme. În recunoașterea modelelor, mulți algoritmi calculează integrale pe suprafața unei regiuni și necesită cunoștințe despre interior. În fotocompunere, fonturile sunt adesea descrise în termeni de contururi care sunt apoi completate pentru a produce copia finală. Problema poate fi rezolvată în mai multe moduri, care pot fi împărțite în două clase mari. În primul, se are o descriere precisă a conturului ca poligon și se decide care părți ale planului se află în interior luând în considerare, de fapt, ecuațiile de linii. Astfel de tehnici ar putea fi numite bazate pe poligon, dar preferăm umplerea marginilor pe termen mai scurt și le descriem în Secțiunea 8.2. Metodele din clasa a doua cartografiază conturul pe planul discret și apoi localizează interiorul examinând valorile pixelilor. Astfel de tehnici bazate pe pixeli sunt discutate în Secțiunile 8.3 până la 8.5.

Pe lângă cele de mai sus, tehnicile pot fi distinse prin principiul pe care îl folosesc pentru a decide dacă un punct se află în interiorul poligonului. Unii algoritmi folosesc verificarea parității, în timp ce alții folosesc un criteriu de conectivitate.

Algoritmii de verificare a parității se bazează pe faptul că o linie dreaptă intersectează orice curbă închisă (cum ar fi conturul unei regiuni) și

numărul de ori (vezi Figura 8.1). Dacă știm că primul punct al dreptei se află în afara regiunii, atunci îl putem traversa și decide care segmente sunt în interior numărând numărul de intersecții. Dacă numărul este impar, atunci segmentul aparține interiorului *(AB* și *CD* în Figura 8.1), altfel nu *(BC* în Figura 8.1). Toți algoritmii de umplere a marginilor folosesc verificarea parității. Principiul poate fi folosit și de algoritmi bazați pe pixeli, dar cineva se confruntă cu o problemă serioasă. Este posibil să aveți puncte din două sau mai multe laturi mapate pe același pixel (vezi Secțiunea 7.6), iar acest lucru produce o numărare incorectă a numărului de intersecții. De asemenea, trebuie să fiți atenți la liniile care sunt tangente la contur: punctul de contact trebuie numărat de două ori ca intersecție. Vă sugerăm modalități de a depăși aceste probleme, cel puțin parțial, în Secțiunea 8.3.

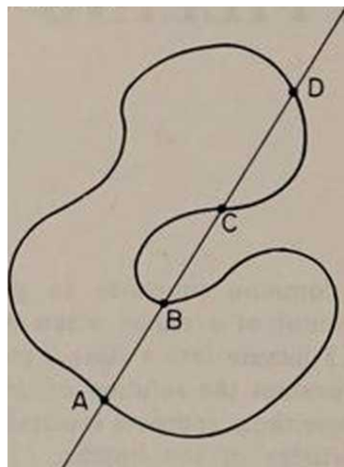


Figura 8.1 Ilustrarea principiului de paritate pentru a decide care puncte aparțin interiorului unei curbe închise

Umplerea conectivității presupune că este dat un punct interior (sământă), în plus față de contur. Se efectuează apoi o traversare a planului pentru a găsi toți pixelii la care se poate ajunge din sământă fără a traversa conturul. Procesul poate fi modelat ca o traversare a graficului folosind unele dintre structurile de date discutate în Capitolul 6. Umplerea conectivității este adecvată numai pentru algoritmii bazați pe pixeli, deoarece necesită acces aleatoriu la pixelii din interior. Umplerea se face cel mai bine pe memoria de reîmprospătare a unui dispozitiv grafic raster sau copia exactă a acestuia în memoria principală. Avantajul major al umplerii conectivității este că este robustă în ceea ce privește neregularitățile conturului, atâta timp cât conturul este o curbă închisă. Dezavantajul său major este necesitatea de a cunoaște un punct interior
avans. Prin urmare, este foarte potrivit pentru grafica interactivă în care un utilizator desenează un contur (posibil destul de neregulat) și apoi indică interiorul acestuia (furnând astfel sământa) și solicită ca acesta să fie umplut. Discutăm despre algoritmi bazați pe conectivitate în Secțiunea 8.4.

Este posibil să combinați conectivitatea și completarea parității în diferite moduri și să veniți cu noi algoritmi. Astfel de combinații sunt implementate cu ușurință dacă folosim o structură de date comună. Se pare că graficul de adiacență a liniilor pentru contur (C-LAG) este potrivit pentru ambele. În mod clar, este justificat pentru un algoritm de verificare a parității bazat pe pixeli, unde accentul este pus pe contur. În timp ce un grafic de linie de adiacență pentru interior (I-LAG) este structura de date folosită în mod obișnuit pentru algoritmii de conectivitate , Propunerea 6.3 sugerează o modalitate de utilizare a C-LAG pentru aceștia. Să presupunem d-conectivitate pentru interior, i-conectivitate pentru contur, culoarea *X* pentru interior și culoarea *Y* pentru pixelii conturului. Atunci Propunerea 6.3 devine:

Propunerea 8.1: Dacă un nod al I-LAG are gradul *(m,n)* cu «>1 atunci va exista un nod al C-LAG cu gradul (0,J) în linia de mai jos. În mod similar, dacă m>l va exista un nod al C-LAG cu gradul (d,0) deasupra acestuia. Este adevărat și invers. □

Prin urmare, în loc să verificăm gradul nodurilor I-LAG (după cum este necesar pentru o traversare a graficului), putem verifica gradul nodurilor C-LAG. O astfel de alegere are două avantaje practice. Una este că permite combinarea algoritmilor de verificare a parității și de conectivitate. Celălalt este o posibilă creștere a vitezei. Evaluarea gradului unui GAL necesită examinarea a trei linii simultan. Pentru I-LAG aceste linii pot fi destul de lungi, în timp ce intervalele de contur tind să fie mult mai scurte. Datorită posibilității de combinații, vom folosi C-LAG pentru ambele tipuri de algoritmi.

8.2 Umplutura marginilor

Umplerea bazată pe margini sau poligoane se mai numește și *umplerea conversiei de scanare,* deși toți acești termeni sunt uneori folosiți în alte scopuri. Laturile poligonului, sau marginile, sunt sortate și marcate astfel încât să păstreze toate informațiile topologice. Apoi verificarea parității este efectuată printr-un algoritm simplu, cu o singură trecere, iar umplerea efectivă nu necesită acces aleatoriu la pixeli. Se poate face la fel de bine într-un dispozitiv de grafică raster sau un dispozitiv de grafică vectorială. Este deosebit de potrivit pentru aplicațiile în care același contur este afișat în mod repetat, cum ar fi în fotocompunerea, deoarece presortarea și marcarea, costisitoare din punct de vedere al calculului, trebuie efectuate o singură dată.

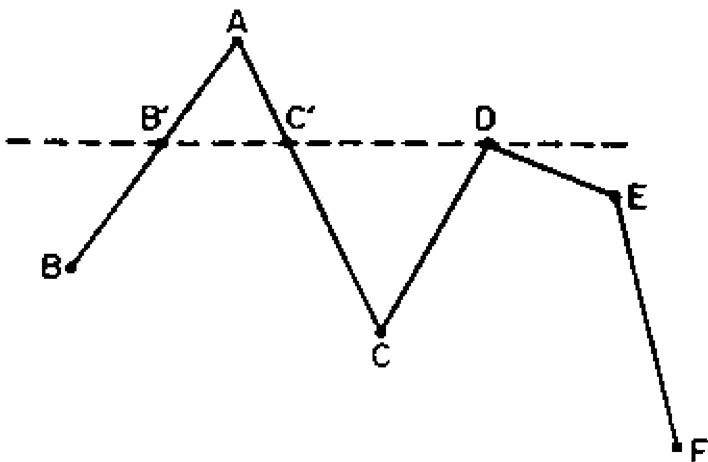


Figura 8.2 **Umplerea marginilor: în cadrul fiecărei felii orizontale există un număr par de margini, iar interiorul regiunii este între prima și a doua, a treia și a patra etc.**

Începem cu succesiunea laturilor (marchiilor) unui poligon care aproximează conturul. Pentru fiecare latură ni se dau coordonatele *Xi^* *t* ale colțului cu maxim y, sau minim x dacă linia este orizontală , iar diferențele Ax,Ay. care dau coordonatele celuilalt colt daca se adauga la primul. Un algoritm de umplere a marginilor procedează prin sortarea mai întâi a laturilor în funcție de valoarea *y* și valoarea x pentru muchiile cu același *y.* Dacă mai există egalitate. Se folosește Axe și, în final, Ay. Când sortarea se face în funcție de aceste diferențe, cel mai mic merge primul. Astfel, în exemplul din figura 8.2, ^S va precede *AC* deoarece Ax este negativ pentru *AB* și pozitiv pentru *AC.* Apoi, algoritmul găsește maximele în direcția lor și detectează intersecțiile muchiilor cu linii paralele cu axa x. În exemplul din figura 8.2 linia prin *D* produce două astfel de intersecții, S' și *C.* Aceste linii împart imaginea în felii cu proprietatea că fiecare felie conține un număr par de margini și interiorul regiunii se află între marginile cu numere impar și par.

Marginile sortate sunt plasate intr-o lista de asteptare iar umplerea se face prin mentinerea unei liste active care contine marginile prezente in fiecare felie. Procesul începe de la cea mai mare valoare a lui y și continuă spre cea mai mică. În lista de așteptare a marginilor sunt inserate diverse steaguri pentru a indica următoarele circumstanțe.

t intr-un itnie algebric. adică toate valorile native sunt considerate mai mici decât toate pozitive.

(a) Câte muchii să transferăm în continuare: una dacă suntem de-a lungul unei laturi, două dacă suntem la maxim, eventual mai multe dacă avem mai mult de un maxim pentru aceeași valoare a lui *y.* (Acest lucru nu este atât de puțin probabil pe cât

pare. De exemplu, un contur al literei W pentru anumite fonturi are trei maxime pentru aceleași.)

(b) Dacă o nouă pereche de muchii trebuie citită înainte de a ajunge la sfârșitul perechii curente. Acesta va fi cazul când linia prin maxim intersectează una dintre muchiile curente.

(c) Fie că la capătul acestei margini nu ar trebui să mai citim pentru că corespunde unui minim în direcția lor.

Pentru exemplul din figura 8.2. prima pereche de muchii se citește *{AB* și *AC),* împreună cu valoarea y, y^, a oricăror maxime din aceeași felie *(D).* Regiunea dintre muchiile cu *y* mai mare decât y $_m$ „ *{B'AC')* este umplută și apoi se citește următoarea pereche de muchii *(CD* și *DE)* Acum avem o listă activă de patru margini și regiunile dintre prima și a doua și a patra dintre ele sunt identificate în mod clar ca atare și nu există nicio problemă cu numărul de paritate acolo, în contrast cu umplerea maximă din lista sortată *(EF)* și se adaugă la lista activă. Cu toate acestea, ar trebui să distingem cazul în care două margini se termină simultan *(AC* și *DC)* și nu este nevoie să adăugați elemente noi la lista activă.

Tabelul 8.1: Descrierea muchiei pentru conturul din Figura 8.2

Latura	X	*y*	Topor	*^y*	Steagul (a)	Hag(b)	Hag (c)
AB	^	*yA*	$x_a - x_A$	y_a *-yA*	2	*yo*	1
AC	X_A	*yA*	X_C~X_A	*yc-yA*	1	•	0
DC	*$_D$	y_D	$x_c - x_D$	*yc-yn*	2	•	0
DE	X_D	*yo*	X_E~X_D	*da-yo*	1	-	1
EF	X_E	*voi*	$x_F - X_E$	*da-da*	1	•	1

Forma listei de așteptare în cazul figurii 8.2 este dată în tabelul 8.1. Indicele corespund etichetelor punctelor din figură. Valoarea dată sub pavilionul (a) este numărul de margini care trebuie transferate simultan din lista de așteptare în lista activă: 2 înseamnă marginea curentă plus următoarea și I înseamnă doar marginea curentă. Flag (b) dă valoarea *y* unde o nouă pereche de muchii trebuie mutată în lista activă. Flag (c) indică numărul de noi muchii care trebuie citite la sfârșitul muchiei curente.

Algoritmul 8.1 Umplerea marginilor

Notație: **Fiecare latură este specificată de valorile *y* și *x* ale celui mai înalt punct final, cu excepția cazului în care este orizontală. Apoi sunt selectate valorile (x,^) ale punctului său cel mai din stânga. *Ax* și *Ay* sunt diferența dintre coordonatele celui de-al doilea punct și cel ales pentru a reprezenta dreapta. Indicele *i* se referă la indexarea originală a laturilor. Etichetele părților laterale sunt aceleași cu steagurile din Tabelul 8.1.**

1. Sortați toate părțile în funcție de valoarea *y* ((mai întâi mai mare). Rupeți legăturile sortând mai întâi după valoarea lui *Xi* și apoi după Ax și Ay, valorile negative precedând pozitiv. {Pregătirea listei de așteptare.}
2. **Pentru** toate perechile de laturi succesive care au același y **face:** Begin.
3. Desenați o linie prin punctul în care astfel de muchii se întâlnesc și găsiți intersecția acesteia cu toate celelalte laturi. {În mod clar, trebuie luate în considerare doar părțile care vin înainte în listă.}
4. Pentru prima muchie intersectată, etichetați intrarea acesteia în lista de așteptare cu valoarea y unde are loc intersecția.

   Sfârşit. {Lista de așteptare este acum gata.}
5. În timp ce lista de așteptare nu este goală, faceți:

   ÎNCEPE.
6. Dacă lista activă este goală sau valoarea lui y este sub punctul final inferior al unuia dintre membrii săi, atunci eliminați acești membri și transferați din lista de așteptare în lista activă câte elemente sunt indicate de etichete.
7. Umpleți spațiul dintre membrii alternativi ai listei active.
8. Decrementare^.

   Sfârşit.
9. **Sfârșitul algoritmului.**

În mod clar, formatul tabelului 8.1 conține informații redundante. De exemplu, s-ar putea renunța la aproape toate valorile lui *Xi* și *y t* (vezi problema 8.2). Nu discutăm aici problema codificării pentru a nu ascunde forma practic simplă a algoritmului. Procesul este descris în detaliu ca algoritmul 8.1. Sortarea pasului 1 poate fi efectuată prin oricare dintre numeroșii algoritmi disponibili în literatură. Modul în care se realizează ruperea de egalitate garantează că muchiile care se întâlnesc la maximum vor fi stocate în locații succesive, cea având mai întâi panta pozitivă. Pasul 2 se uită la maxime. Deoarece valoarea *y* asociată fiecărei muchii este cea a punctului final corespunzător celei mai mari dintre cele două valori, egalitatea lui *y* în două muchii succesive înseamnă un maxim. Datorită regulii de spargere a egalității în sortare, un argument geometric simplu poate fi folosit pentru a arăta că nu trebuie să verificați valoarea x. Pașii de la 1 la 4 pregătesc lista de așteptare și trebuie efectuate o singură dată pentru fiecare contur. Pașii de la 5 la 9 fac umplerea propriu-zisă.

Algoritmul nu face distincție între contururile găurilor și contururile externe și nu se limitează la doar regiunile conectate. Va umple corect grupuri de regiuni dacă contururile acestora au fost sortate împreună. (De exemplu, va umple conturul literei „i”.)

Descrierea dată aici nu prevede cazul în care părțile conturului sunt linii orizontale. Cu toate acestea, algoritmul este ușor de modificat pentru a gestiona corect astfel de contururi (vezi problema 8.1).

8.3 UMPLUREA CONTURULUI PRIN VERIFICAREA PARITĂȚII

Deși implementarea unei verificări de paritate este trivială într-un algoritm de umplere a marginilor, este netrivială în planul discret. Pe lângă dificultatea identificării tangentelor, care trebuie socotite ca puncte duble, trebuie să vă faceți griji cu privire la mai mulți pixeli care pot elimina numărul de intersecții. Dificultățile de a defini liniile și intersecțiile liniilor și curbelor detaliate în Capitolul 7 ar trebui să fie un avertisment cu privire la riscurile abordării. Vom prezenta aici un algoritm care utilizează C-LAG și apoi vom arăta că pentru regiuni întregi dă rezultate corecte. Algoritmul 8.2 arată funcționarea de bază a unei verificări de paritate.

Algoritmul 8.2 Trivia! Verificarea parității în avion

0. Stabiliți un sistem de coordonate *xy* .
1. **Pentru** fiecare *y* **faceți** pașii 2-6.

   ÎNCEPE.
2. Setați *numărul* egal cu zero.
3. Pentru fiecare x de la stânga la dreapta, faceți pașii 4 și 5.

   ÎNCEPE.
4. Dacă (x ^) aparține conturului, atunci incrementează *numărul.*
5. **În rest, dacă** *numărul* este impar, **atunci** pixelul (x,y) este în interior a regiunii.

   Sfârşit.

   Sfârşit.
6. **Sfârșitul algoritmului.**

În mod clar, algoritmul 8.2 este incorect. În primul rând, numără pixelii mai degrabă decât intervalele de pixeli adiacenți care au culoarea conturului. (Reamintim că pe planul discret intersecția a două curbe nu este întotdeauna un singur pixel.) În al doilea rând, nu caută extreme. Putem dezvolta un algoritm corect, cel puțin pentru regiuni întregi, dacă numărăm mai degrabă intervalele decât pixelii și dacă examinăm liniile de deasupra și dedesubtul celei utilizate pentru testul de paritate. Dacă unul dintre ele nu intersectează conturul în

vecinătate a intersecției curente, atunci știm că linia curentă este o tangentă.

Algoritm 8.3 bazat pe pixeli, algoritm de umplere a verificării parității.

Notație: count **este numărul de intersecții ale conturului cu linia orizontală curentă. Variabilele** *de mai sus* **și** *de dedesubt* **sunt gradele C-LAG returnate de** *LINK.*

1. **Pentru** fiecare *y* **faceți** pașii 2-1 L

ÎNCEPE.

2. Setați *numărul* la zero

3. Setați x la valoarea cea mai din stânga a grilei.

4. **În timp ce x** este mai mic sau egal cu valoarea din dreapta a lui grila **faceți** pașii 5-11.

ÎNCEPE.

5. **Dacă** (x,y) nu aparține conturului, **atunci faceți** pașii 6 și 7.

ÎNCEPE.

6. Dacă *numărul* este impar, atunci marcați (xy) ca aparținând interiorul.

7. Creșteți x.

Sfârşit.

8. **În rest, faceți** pașii 9-11.

ÎNCEPE.

9. **Procedura** de apel *LINK.*

1°- **Dacă** ambele *de deasupra* și *dedesubt* sunt egale cu 1, **atunci** crește *conta.*

IL **Dacă** suma *de mai sus* + *de dedesubt* nu este 0 sau 2, **atunci** setați indicatorul de eroare.

Sfârşit.

Sfârşit.

Sfârşit. **. . .**

12. **Sfârșitul algoritmului.**

Graficul de adiacență a liniilor de contur (C-LAG) oferă o structură de date convenabilă: nodurile sale corespund intersecțiilor conturului cu o linie orizontală. Pentru un maxim gradul *de deasupra* este zero, iar pentru un minim gradul *de mai jos* este zero. Dacă conturul nu are mai multe puncte, atunci va avea o intersecție a unui

arc cu o linie orizontală
gradul (1,1) în C-LAG. În plus, extrema va avea grade (0,2) sau (2,0). Toate celelalte perechi de grade sunt configurații ilegale dacă regiunea este plină. Algoritmul 8.3 implementează aceste idei. Utilizează procedura LINK *(* listată ca algoritmul 8.3a) pentru a găsi gradele nodurilor C-LAG prin numărarea numărului de intervale care se suprapun cu cel curent. De asemenea, presupune că x și y sunt măsurate în unități ale dimensiunii celulei grilei. Indicatorul de eroare din lista algoritmului semnifică apariția altor grade decât perechile legale (1,1), (0,2) și (2,0). Figura 8.3 prezintă câteva exemple de valori ale gradelor LAG pentru diferite configurații de pixeli.

Algoritmul 8.3a Găsirea gradului unui nod în LAG

Procedura *UNK*

Natation: dy **este creșterea valorii lui** *y* **de la o linie la alta. Dacă originea coordonatelor este în colțul din stânga jos al afișajului, atunci** *dy* **este pozitiv. În caz contrar, este negativ,** *dx* **este incrementul în valorile lui** *x.* **De obicei, ambele** *dx* **și** *dy* **sunt egale cu unul.**

0. Intrarea este pixel pe contur (xy). Ieșirea este valorile de *mai sus* și *de mai jos.*
1. Setați contoarele *de deasupra* și *dedesubt* la zero.
2. **dacă** *(x—dx,y+dy)* aparține conturului, **atunci** creșteți *mai sus.*
3. **dacă** *(x—dx j—dy)* aparține conturului, **incrementul de** mai *jos.*
4. **În timp ce** (x *y*) aparține conturului, **faceți** pașii 5-7.

   ÎNCEPE.
5. **Dacă** *(x^+dy)* aparține conturului și *(x~dx j+dy)* aparține nu, **apoi** crește *mai sus.*
6. dacă *(xj—dy)* aparține conturului și *(x—dx,y—dy)* nu, **atunci** creșteți *mai jos.*
7. Creșteți x.

   Sfârşit.
8. Dacă *(x-dx j+dy)* nu aparține conturului și *(x^+dy)* face, atunci creșteți *mai sus.*
9. dacă *(x—dx^—dy)* nu aparține conturului și *(xjf—dy)* face, **atunci** creșteți *mai jos.*
10. Returnează locația pixelului *(x,y)* și valorile contoarelor *de deasupra* și *dedesubt.*
11. **Sfârșitul algoritmului.**

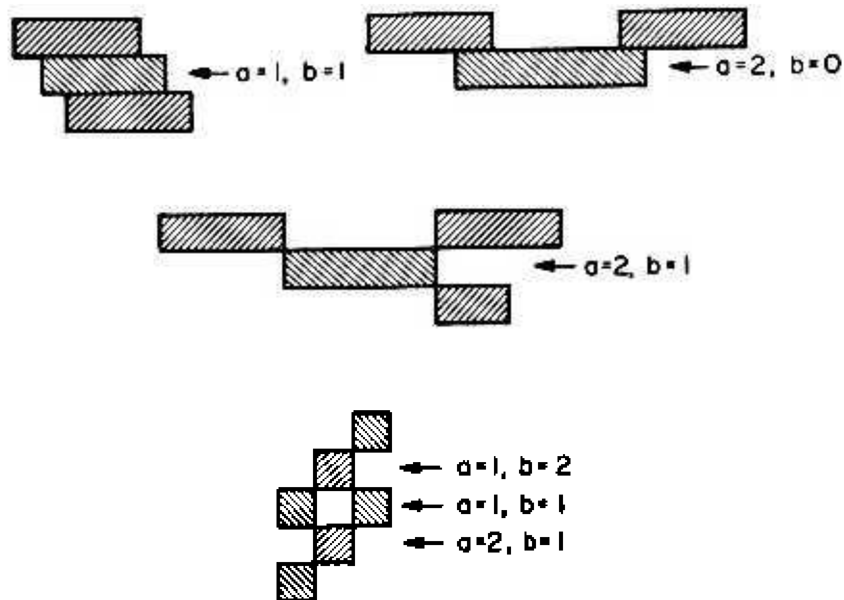


Figura 8.3 Exemple de configurații de suprapuneri de intervale cu valorile contoarelor utilizate în algoritmul 8J/3a. *a* **reprezintă** *mai sus* **și 6 pentru** *dedesubt.*

8.3.1 Dovada corectitudinii algoritmului 8.3

Trebuie să studiem condițiile în care este setat pavilionul de eroare. Se pare că acest lucru se poate întâmpla numai în jurul mai multor pixeli. Deoarece o limită este o curbă închisă, este adevărat că va intersecta un set plan convex (de exemplu, celula corespunzătoare unui pixel) de un număr par de ori. Un pixel simplu conține doar un astfel de arc și, prin urmare, ar trebui să aibă exact doi vecini pe contur. Același lucru va fi valabil și pentru un set de pixeli simpli și, în special, pentru un set de pixeli adiacenți de-a lungul liniei orizontale. Pe de altă parte, un pixel multiplu poate avea mai mult de doi vecini pe contur sau doar unul. Atunci urmărirea conturului poate să nu corespundă cu urmărirea limitei. Situația se complică și mai mult deoarece ne ocupăm de intervale mai degrabă decât de pixeli unici. Figura 8.4 prezintă un contur cu un număr de pixeli multipli în care numărul de paritate nu este corect, ceea ce face ca indicatorul de eroare să fie setat. Tabelul 8.2 listează valorile gradelor nodurilor LAG corespunzătoare și modificările corespunzătoare ale *numărului* dacă se dorește să se facă corect umplerea.

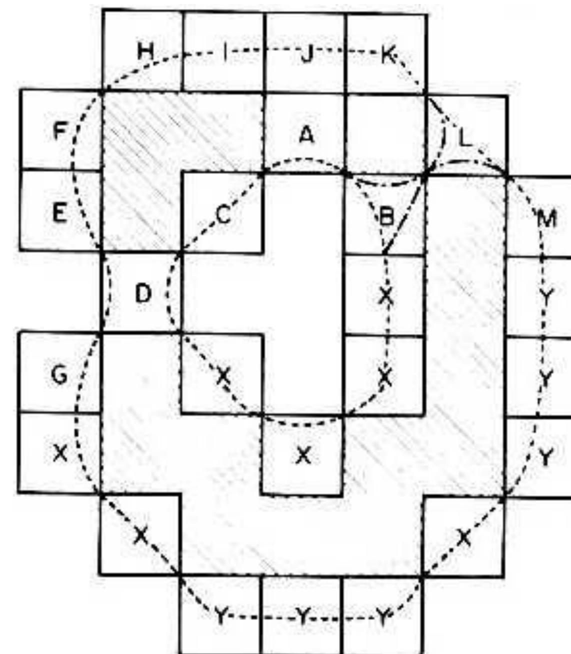


Figura 8.4 Interpretări ale trasării limitelor corespunzătoare unui pixel cu trei vecini. Pixelii din interior sunt afișați umbriți.

La prima vedere, s-ar putea părea că pixelii *B* și *L* nu sunt multipli, conform definițiilor din Capitolul 7. Cu toate acestea, este posibil să existe un arc de contur prin pixelii *B, L, K* $_t$ și *A* și un alt arc prin *X, B, L, M, Y,* • • • , etc. O situație ambiguă similară a fost prezentată în Figura 7.15b și a motivat cerința ca regiunile d-conectate să aibă un interior. Trebuie menționat că un pixel multiplu nu provoacă întotdeauna condiția de eroare.

Tabelul 8.2: Intervale de probleme din Figura 8.4

Interval	grad	Numărul potrivit
(HK)	(0,3)	0
(O)	(1,2)	0
(D)	(2,2)	2
(B)	(2.1)	1
(L)	(1,2)	1

Aceste exemple ilustrează un caz destul de extrem și în multe aplicații grila de eșantionare este suficient de fină încât să nu apară mai mulți pixeli. Vom arăta că în astfel de condiții algoritmul 8.3 este corect.

Teorema 8.1: Dacă o regiune este plină, atunci singurele perechi de grade posibile pentru conturul ei sunt (0,2). (2.0) și (1,1).

Demonstrație: Demonstrăm teorema arătând imposibilitatea apariției altor perechi. În primul rând, examinăm cazul unei perechi de grade (0,1). Dacă intervalul curent și cel de dedesubt conțin fiecare un singur pixel, atunci avem un pixel ale cărui c-vecini coincid (Figura 8.5a). Dacă intervalul de mai jos are doi pixeli, atunci aceștia sunt d-vecini unul celuilalt (Figura 8.5b). Dacă are mai mult de doi pixeli, atunci unii dintre ei sunt d-vecini ai pixelilor de contur din linia de mai sus (Figura 8.5c). Ultimele două situații apar indiferent de numărul de pixeli din intervalul curent. Toate cele trei configurații încalcă condițiile

Definiția 7.6 și, prin urmare, nu poate apărea atunci când regiunea este plină. Cazul (1.0) este tratat în mod similar. Trebuie să arătăm acum că, dacă regiunea este plină,

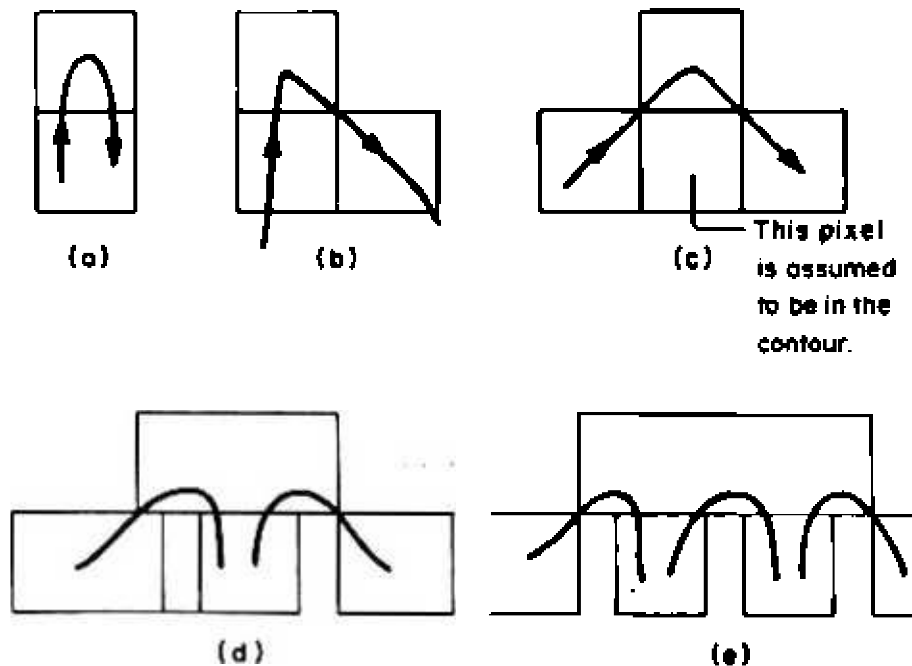


Figure 8.5 **Illustration used in the proof of Theorem 8.1: (a) (0.1) degree with one pixel in the line below; (b) (0,1) degree with two pixels in the line below; (c) (0.1) degree with three pixels in the line below; (d) (0,3) degree; and (e) (0.4) degree.**

niciunul dintre grade nu poate depăși 2. În primul rând, dacă gradul este impar, înseamnă că una dintre ramuri este parcursă cu multiplicare (Figura 8.5d). Dacă gradul este egal, atunci unii pixeli trebuie să aibă mai mult de doi d-vecini în cale (Figura 8.5e). □

Corolar: Dacă algoritmul 8.3 examinează conturul unei regiuni întregi, atunci indicatorul de eroare (pasul 11) nu este niciodată setat. □

Prin urmare, contururile regiunilor complete pot fi completate cu un algoritm relativ simplu, examinând doar câteva linii la un moment dat, dacă regiunea nu este plină, atunci un astfel de algoritm poate eșua, așa cum se arată în Figura 8.4 și Tabelul 8.2. Din păcate, pentru regiunile care nu sunt pline, algoritmul nu numai că eșuează, dar o face fără a seta indicatorul de eroare. Figura 8.6 prezintă un astfel de exemplu. Acolo, nodul superior are gradul (0,2), dar intervalul corespunzător conține patru arce de limită. Rețineți că nu există nicio modalitate ca un algoritm de verificare a parității bazat pe pixeli să umple corect un astfel de contur, cu excepția cazului în care traversează conturul de mai multe ori. Este posibil să umplem corect o regiune arbitrară *R* dacă luăm în considerare conturul lui *R* ca o regiune și îl parcurgem pentru a găsi un nou contur în timp ce se marchează pixelii cu multiplicitatea lor (vezi Notele bibliografice).

8.3.2 Implementarea unui algoritm de verificare a parității

O verificare de paritate bazată pe pixeli are anumite avantaje față de o umplere a marginilor, deoarece atât sortarea în funcție de y. iar interpolarea între punctele date

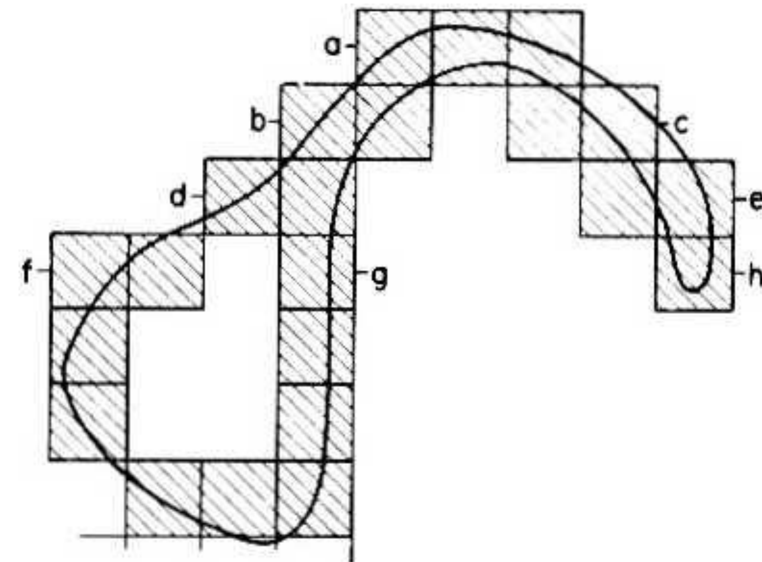


Figure 8.6 **A contour that cannot be filled properly by Algorithm 8.3. Furthermore, the error flag is not set until after some erroneous filling: the pixel between intervals *b* and c.**

(dacă există lacune în valorile lui) nu se fac în computerul gazdă, ci pe dispozitivul de afișare. Mai întâi se poate afișa conturul folosind o comandă vec(xj') (vezi Secțiunea 1.7, Tabelul 1.2). În acest fel, interpolarea este realizată de hardware-ul dispozitivului fără niciun cost pentru computerul gazdă. Acest lucru realizează, de asemenea, sortarea corectă a punctelor, deoarece procesul de afișare poate fi considerat o „sortare a găleților". Două opțiuni sunt apoi deschise. Într-una, algoritmul poate fi implementat în microprocesorul care controlează dispozitivul de afișare . În acest caz, grila este memoria de reîmprospătare. În cealaltă, un grup de trei linii sunt citite în memoria gazdă folosind o instrucțiune *rpic* (vezi Secțiunea 1.7, Tabelul 1.2). Verificarea parității este efectuată acolo și apoi linia de mijloc este scrisă înapoi. Prețul pe care îl plătiți pentru această comoditate este că numai regiunile complete sunt completate corect.

8.4 UMPLUREA CONTURULUI DUPĂ CONECTIVITATE

Dacă cunoaștem un punct din interior, atunci îl putem folosi ca *sămânță* și îi putem propaga culoarea la pixelii adiacenți până când se ajunge la conturul. Algoritmul poate fi descris recursiv în termeni simpli: Fie *c* culoarea seminței *S* și *F* regiunea umplută la un pas. Setăm inițial F = {S}, iar apoi la fiecare pas adăugăm la *F* toți pixelii a căror culoare este *c* și care au un vecin direct în *F.* Astfel, umplerea conturului se realizează pe baza conectivității. Numele „algoritm de însămânțare" este, de asemenea, folosit pentru a descrie un astfel de proces. Ipoteza despre cunoașterea unei semințe este realistă în sistemele grafice interactive, în care un utilizator poate îndrepta un pix sau un cursor către interiorul unui contur. Este valabil și în sistemele în care un solid este definit de vârfurile unui poliedru de aproximare și diferite vederi sunt afișate pe ecranul grafic. Punctele interioare ale diferitelor poligoane rezultate pot fi calculate cu ușurință. În sistemele de procesare a imaginii, un punct din interior poate fi găsit ca unul direct adiacent unui pixel de contur și poate fi marcat atunci când conturul este extras. O verificare de paritate parțială poate fi folosită și ca preprocesor.

Umplerea conectivității este mai puțin probabil să producă deversări decât umplerea bazată pe verificarea parității. Într-adevăr, culoarea actuală a interiorului este cunoscută și un pixel nu este umplut decât dacă este de acea culoare. Spillover-ul poate apărea numai atunci când conturul dat nu este închis. Prețul pe care îl plătim pentru asigurarea împotriva răspândirii este că regiunile care au mai mult de o componentă necesită o sămânță pentru fiecare componentă.

8.4.1 Umplere recursiva a conectivității

Interiorul poate fi parcurs în mai multe moduri folosind structurile de date din Capitolul 6. Cel mai simplu algoritm în ceea ce privește lungimea programului folosește grila de pixeli și se bazează pe recursivitate. Este listat ca algoritmul 8.4. Se examinează cei patru d-vecini ai seminței, iar pentru fiecare dintre ei care nu aparține conturului numim din nou *FILL* cu acel vecin ca sămânță. O ușoară modificare ar putea face să umple orice pixel cu culoarea seminței.

Algoritmul 8.4 Umplere recursiva

Notație: sămânța **este pixelul dat și C este culoarea umpluturii. Pixelii contour au deja această valoare. Variabila** p_N **denotă** *N*- **vecinul** *lui p,* **0 < *N* <7 (vezi Definiția 7.1).**

1. **Apelați** *FILL (sămânță).*
2. **Sfârșitul algoritmului.**

Procedura FILL(p)

Fl. Setați culoarea lui *p* la *C.*
F2. **Pentru** *N* = 0, 2, 4, 6 **faceți:**
ÎNCEPE.
F3. Dacă culoarea lui p_N nu este **C, atunci apelați** *FILL*(p_N). **Sfârșit.**
F4. Reveni

Simplitatea aparentă a acestui algoritm este înșelătoare, deoarece de fiecare dată când este apelată o subrutină există un anumit cost de calcul care nu este evident în listă. Vom reveni la acest punct în secțiunea 8.5.

8.4.2 Umplere nerecursivă de conectivitate

În loc să trecem toată contabilitatea traversării către sistemul care gestionează apelurile subrutinei, am putea dori să facem asta noi înșine, mai eficient. Pentru grafica raster, LAG-ul este o structură de date deosebit de convenabilă. În forma sa cea mai simplă, algoritmul pornește de la sămânță și continuă spre stânga până când este găsit un pixel de contur. Apoi se repetă o scanare spre dreapta până când conturul este văzut din nou. Apoi linia următoare este umplută și așa mai departe până când se ajunge la partea cea mai inferioară a conturului. Procesul se repetă pentru partea din regiunea de deasupra pixelului de pornire. Acesta este un proces destul de simplu, dar se poate observa cu ușurință că umple un contur cu succes numai dacă regiunea este convexă. Poate fi modificat pentru a umple regiuni neconvexe prin utilizarea unei stive în care sunt plasați pixeli aproape de maximele sau minimele conturului. Ulterior, acestea

pixelii sunt scoși din slăbire și folosiți ca semințe. În exemplul din Figura 8.7, pixelii + , B și C ar fi plasați toți în stivă în timp ce partea umbrită este umplută. Pixelii *D* și *E* vor fi plasați acolo ulterior, în timpul parcurgerii părții neumbrite. Deși ideea de bază este destul de simplă, implementarea ei corectă necesită grijă.

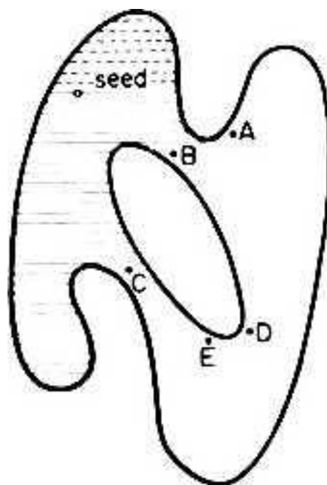
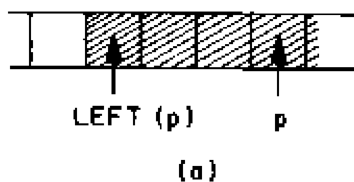


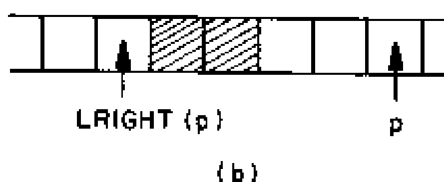
Figura 8.7 **Umplerea conectivității pentru o regiune neconvexă. Pentru sămânța dată, se continuă în jos umplerea părții umbrite de linii orizontale complete. Apoi se întoarce la sămânță și merge în sus umplând partea umbrită cu linii întrerupte.**

Înainte de a descrie algoritmul principal, schițăm anumite proceduri utilizate de acesta.

8.4.3 Proceduri utilizate pentru umplerea conectivității

Vom folosi termenul adresa pixelului pentru a desemna descrierea locației pixelului. Aceasta poate fi o pereche de coordonate *xy* , un pointer către o matrice etc. Dacă *p* este adresa pixelului A, expresia p—1 indică adresa pixelului din stânga acestuia (pe aceeași linie orizontală) și p+1 adresa pixelului din dreapta acestuia. Una dintre aceste adrese nu este definită pentru pixelii din dreapta și alta pentru pixelii din stânga fiecărei linii. Orice implementare a algoritmilor trebuie să facă prevederi speciale pentru astfel de pixeli, dar nu dorim să ne complicăm descrierea cu astfel de cazuri speciale.





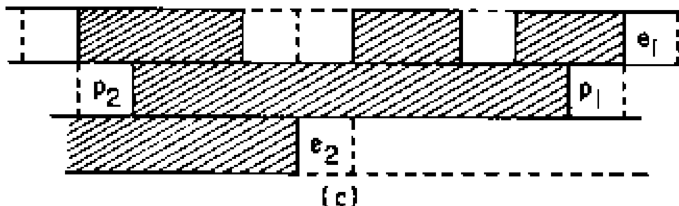


Figura 8.8 (a) Definiția *LEFT(p).* **(b) Definiția** *LRIGHT(p).* **(c) Definiția pixelilor** p_x, p_2, **e>. si ej. Argumentul p** *al* **procedurii** *LINK* **este orice pixel din zona umbrită între** p_2 **și** p_x.

Procedura *LEFT(p)* returnează adresa pixelului din stânga al intervalului de culoare fix care conține *p* (Figura 8.8a). Procedura *LRIGHT(p)* este definită ca

LR!GHT(p) = STÂNGA (STÂNGA (p)-\)-\,

adică, returnează adresa pixelului din dreapta la stânga lui *p* care are aceeași culoare ca *p* și are proprietatea că există cel puțin un pixel de culoare diferită între cei doi (Figura 8.8b). Implementarea unor astfel de proceduri este simplă.

Gradele unui nod (interval) al LAG-ului care conține pixelul *p* sunt găsite prin procedura *LINK(p)* dată ca algoritmul 8.3a. Este necesară o ușoară modificare pentru a specifica adresele pix els «] și ?2 prezentate în Figura 8.8c. Dacă oricare dintre acești pixeli nu este definit, adresa returnată este zero. (Dacă zero este o adresă legitimă, atunci implementatorul poate alege o altă valoare pentru a inițializa fj și r_2 în *LINK.)*

Deoarece sămânța poate fi dată în mijlocul unei regiuni, este necesar să faceți umplerea în ambele direcții de la aceasta. așa cum se arată în Figura 8.7. O astfel de traversare bidirecțională nu este necesară pentru niciun alt pixel din stivă. În loc să aveți o procedură specială pentru parcurgerea primei regiuni, cel mai bine este să acordați o oarecare atenție inițializării. Dacă pixelul *seedg^.* direct deasupra semințelor, nu este în contur, apoi *LEFT (semințele la*»,)* ar trebui să fie plasate în stivă cu indicația că ar trebui să fie folosită pentru a începe o scanare în sus. Acesta este de așteptat să fie

caz obișnuit în grafica interactivă, deoarece un utilizator este probabil să furnizeze un pixel departe de contur ca sământă. Dacă pixelul de deasupra seminței este pe contur, atunci procedăm după cum urmează. Mai întâi examinăm gradul acelui interval de contur printr-un apel către *LINK.* Dacă are grad (0,2), atunci sământa a fost aleasă chiar sub un maxim, deci nu este nevoie de o scanare în sus. În caz contrar, sământa a fost aleasă chiar sub un arc aproape orizontal al conturului. Apoi înlocuim sământa cu *LEFT(seed)* și apelăm *LINK* pentru intervalul de contur din stânga seminței. Pixelul Cj este apoi ales ca sământă pentru o scanare în sus.

8.4.4 Descrierea algoritmului principal

Algoritmul poate fi descris informal după cum urmează. Umple liniile orizontale mergând de la stânga la dreapta pornind de la un pixel imediat la dreapta conturului. (Acest lucru poate fi găsit în prima linie printr-un apel de *STÂNGA (seed).)* Fie *p* acel pixel. Înainte de completare, se efectuează un apel către *LINK (p—\)* pentru a investiga C-LAG al intervalului de contur stânga. Dacă *deasupra* și *dedesubt* sunt ambele una, atunci avem un arc simplu și specificăm locațiile pentru următoarea linie de scanare. Dacă direcția de scanare verticală are valoarea „în jos”, atunci setăm *p^ = e 2 , în caz* contrar *Pw* „*i- Apoi procedăm, umplând orizontal începând de la *p până* când este *găsit un* pixel de contur Fie p drept *adresa ultimului pixel* interior *.*

Cazurile interesante sunt cele în care *deasupra* și *dedesubtul* nu sunt ambele unul la ambele capete. În special, dacă unul dintre ele este zero, atunci știm că avem un extremum și trebuie fie să plasăm un pixel pe stivă, fie să încheiem umplerea într-o anumită direcție. Aceasta este partea esențială a algoritmului care este descrisă în continuare.

Algoritmul 8.5 se bazează pe aceste idei. Pe lângă stiva *S,* folosește și o stivă S_d pentru a stoca direcția corectă în același timp în care plasează o adresă în *S.* (Desigur, este posibil să se folosească un singur stivă ale cărei elemente sunt o pereche de valori, adresa pixelului și direcția .) *POP (stiva)* este așa cum este listat în algoritmul 6.1. Pasul 3 examinează direcția de scanare și stabilește anumite variabile în consecință. În special , *u* = 1 înseamnă că *e^* va fi selectat ca următorul pixel, în timp ce *u = 2* va determina selectarea e_2. Utilizarea acestor variabile elimină necesitatea de a avea coduri separate pentru parcurgerea în sus și în jos. Bucla constând din pașii 5-17 este partea principală a algoritmului. Pentru a scana partea de sus a imaginii trebuie să plasăm sământa pe stivă de două ori.

Pasul 5 examinează dacă următorul pixel este o adresă legală sau dacă a fost deja completat. În acest caz, algoritmul iese din bucla interioară și execută pasul I. Dacă stiva nu este goală, apare un nou pixel și se repetă întregul proces. Dacă următorul pixel este o adresă legală și nu este încă completat, atunci acesta devine pixelul curent (pasul 6). Pasul 7 evaluează gradele C-LAG la capătul din stânga.

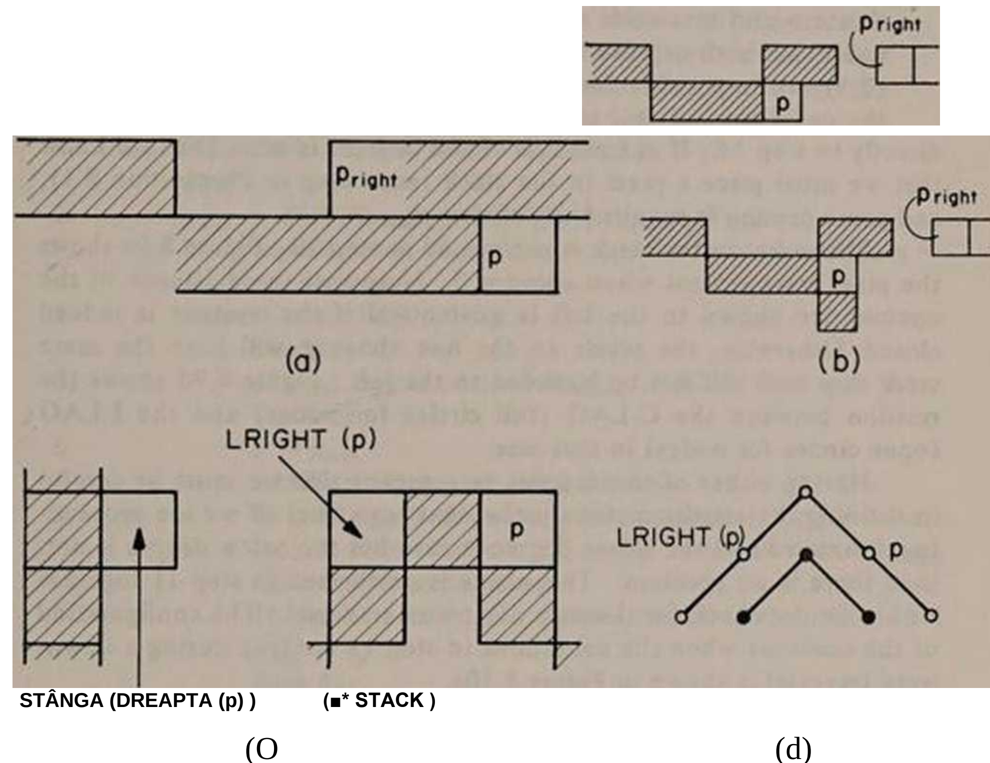


Figura 8.9 (a) și (b) Configurații care fac ca condiția dintre paranteze ((...]) din Pasul 8 al algoritmului 8.5 să fie adevărată, atunci când scanați în direcție în jos. Cu toate acestea, doar (a) indică faptul că a fost atins un fund și umplerea în acea direcție se oprește, (c) Când se examinează conturul din stânga și se găsește configurația de pixeli *LAG* **(să găsească stiva de pixeli, LAG) configurație (c).**

Pasul 8 efectuează prima operație nebanală: verifică dacă am ajuns la capătul interiorului în direcția scanării. Dacă direcția de scanare este în jos și *deasupra* depășește unu, trebuie să fi ajuns fie la o parte de jos, așa cum se arată în Figura 8.9a, fie la una dintre configurațiile prezentate în Figurile 8.9b. Primul caz este detectat prin verificarea faptului că pixelul curent *p* este la dreapta lui *p^.* și prin urmare

în afara conturului. Apoi completarea în direcția respectivă este încheiată și algoritmul iese din buclă. Altfel știm că pixelul actual este într-adevăr în interior. (Dacă direcția de scanare este în sus, avem un test similar pentru a verifica dacă am ajuns la vârf.)

Pasul 9 verifică dacă atât gradele de mai sus, cât și de dedesubt sunt diferite de zero și pune deoparte punctul de pornire în linia următoare. Nu insistăm ca ambele grade să fie unul, chiar dacă nodurile cu gradul (1,2) sau (2,1) sunt imposibile pentru regiunile complete. Pentru majoritatea pixelilor conturului, condițiile acestui pas vor fi adevărate, iar algoritmul va trece direct la pasul 14. Dacă cel puțin unul dintre grade este zero, atunci wc știm că trebuie să plasăm un pixel în stivă (conform Propoziției 8.1) și este necesară o anumită precauție în definirea P_M-

Plasarea în stivă se efectuează în pasul 10. Figura 8.9c arată aranjamentul pixelilor când *este mai sus* = 0. Rețineți că existența arcului de contur arătat în stânga este garantată dacă conturul este într-adevăr închis. În caz contrar, pixelii de pe linia de deasupra *p* vor avea aceeași culoare ca și *p* și nu vor fi mărginiți la stânga. Figura 8.9d arată relația dintre C-LAG (cercurile complete pentru noduri) și I-LAG (cercurile deschise pentru noduri) în acest caz.

A avea oricare dintre grade zero înseamnă că trebuie să fim atenți în definirea punctului de plecare pentru următoarea linie de scanare. Dacă mergem în jos și gradul de mai sus este zero, dar gradul de dedesubt nu este, atunci nu există nicio problemă. Această verificare este efectuată în pasul 11 împreună cu o verificare similară pentru cazul traversării în sus. Configurația contururilor atunci când condițiile din pasul 11 sunt adevărate în timpul unei traversări în jos este prezentată în Figura 8.10a.

Pasul 12 se ocupă de cazul în care nu există pixeli de contur în următoarea linie (în funcție de direcția de scanare) conectați la contur. Această configurație este prezentată în Figura 8.10b pentru o direcție în jos. Examinează pixelul direct deasupra *(p_{ontop})* sau direct sub *($p_{und „}$)* cel curent. Dacă acel pixel nu este umplut, atunci îl selectează ca următor pixel pe aceeași linie care se află imediat în dreapta conturului. (Seci comentariile de la pasul 10 privind existența unui astfel de arc de contur.)

Pasul 14 documentează umplerea efectivă a liniei. Este pasul care vede majoritatea pixelilor. Pentru fiecare pixel verifică dacă pixelul are culoarea conturului, iar dacă nu umple pixelul cu culoarea corespunzătoare.

Algoritmul 8.5 Completare prin conectivitate

Notație: p^ **este pixelul care va fi folosit pentru a începe umplerea pe noua linie,** p^, **poate avea valoarea** *p^o?* **sau** p_{unttf} *și* **este pixelul situat direct deasupra sau direct sub cel curent. S este un teanc de adrese de bază și** S_d **un teanc de direcții pentru parcurgerea LAG-ului.**

0. Intrarea este matricea imaginii, adresa unui *seed interior de pixeli* și, eventual, un al doilea pixel *seed* $_{OMop}$. *STÂNGA (sământa)* este plasată în *S* și direcția *în jos* în S_d. Dacă este dat al doilea pixel, *LEFT (seed* $_{onlop}$*)* este de asemenea plasat în *S* și direcția *sus* în S_d.
1. **În timp ce** stiva *S* nu este goală, **repetați** pașii 2-19 Începeți.
2. P_M = POP(S), *dir* - POP^), *p,^ = %* $_{max}$
3. Dacă *dir* este egal *cu down,* atunci setați u = 2, *other = under.* Altfel *u* = 1, *other = pe deasupra.*
4. **Repetați** pașii 5-16

ÎNCEPE.

5. Dacă P_M este egal cu zero sau dacă P_M este umplut, atunci ieșiți din bucla.
6. *P În continuare*
7. *LINK(p—\)* {verificați conturul stâng}
8. **Dacă** *[(dir* este egal *în jos* și *mai sus* depășește 1) sau *(dir* este egal *sus* și *dedesubt* depășește 1)1 și *p* este la dreapta lui *p* $_{ng}$ *t"* apoi ieși din buclă.
9. **Dacă** atât *deasupra* cât și *dedesubtul* depășesc zero, atunci setați *Next* $^{= e}$ *u* •

Altfel face:

ÎNCEPE.

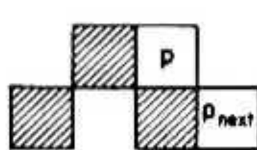
10. **Dacă** *p* nu este deja completat, **atunci** plasați în *S* adresa *LEFT(LRIGHT(p))* iar în S_d valoarea *dir.*
11. **Dacă** *(sus* este egal cu zero și *mai jos* nu și *dir* equals *down)* **sau** *(sub* equals zero and *over* does not and *dir* equals *up),* apoi setați *p^ — e* $_u$.
12. **Altfel, dacă** *Pah"* nu este completat, **atunci** setați *pM* = *STÂNGA (altă)* •
13. **Altfel setați** P_M **= 0-**

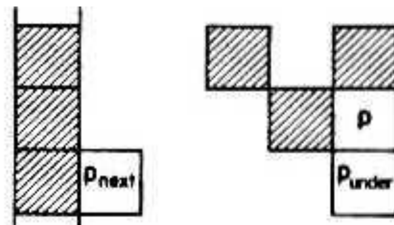
Sfârșit.

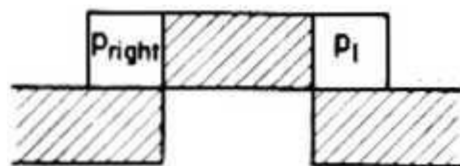
14. Completați linia începând de la *p* și lăsați *pright* să fie ultimul pixel înaintea conturului.
15. *LINK(pri* $_g$ *ht+l)* {verificați conturul din dreapta}

16. **Dacă** deasupra sau *mai jos este egal cu* 0 și **dacă** pi nu este umplut, **atunci fa:**
ÎNCEPE.
17. Locul p| în suge *S*
18. Dacă *peste> 0.* **apoi** plasați în *S^ .*
19. **Altfel** plasează *jos* în *S a .*
Sfârşit.
Sfârşit.
Sfârşit.
20. **Sfârșitul algoritmului.**

Pasul 15 verifică gradul C-LAG la nodul corespunzător arcului de contur din dreapta. Dacă oricare dintre grade este zero, atunci pixelul de pe cealaltă parte a arcului de contur (pj în Figura 8.10c) este plasat în aspirație, împreună cu direcția corespunzătoare. (Pașii de la 15 la 17.) Figura 8.10d arată configurația în termenii celor două LAG-uri.







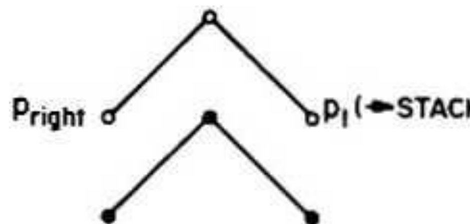


Figure 8.16 (a) Configuration when the conditions of step 11 are true, (b) Configuration when the conditions of step 12 are true, (c) When the right contour is examined, and the configuration shown is found, pixel p is placed in the suck, (d) The two LAGs for configuration (c).

Observăm că cel mult două noduri simultan sunt plasate pe aspirator. Cu toate acestea, toate nodurile conectate la unul dat sunt în cele din urmă fie plasate în aspirator, fie umplute. În exemplul din Figura 8.11, să fie C primul nod vizitat în timpul unei traversări în sus. Apoi *B* și *D* sunt plasați pe aspirator. Ori de câte ori *D* este vizitat din nou (fie fiind tras din sută, fie prin atingerea unui alt nod), £ va fi plasat în suck.

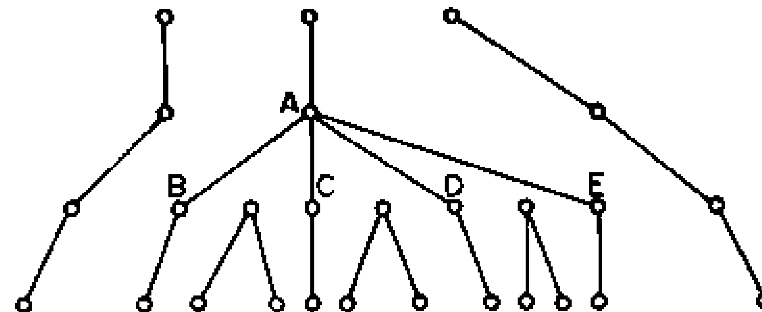


Figura 8.11 Dacă C este primul nod vizitat, nodurile *B* și *D* sunt plasate în stivă. Nodul £ va fi plasat numai atunci când *D* este vizitat din nou.

De asemenea, se poate arăta că pentru o regiune completă Algoritmul 8.5 face umplerea corect. Într-adevăr, în acest caz, singurii pixeli plasați pe aspirator sunt cei care sunt adiacenți maximelor sau minimelor în direcția verticală .

20.5 COMPARAȚII ȘI COMBINAȚII

Dacă comparăm algoritmii bazați pe pixeli în ceea ce privește lungimea programului. Algoritmul 8.4 este în mod clar cel mai simplu, urmat de verificarea parității, Algoritmul 8.3, cu Algoritmul 8.5 ultimul. În anumite privințe, calculele efectuate de ultimii doi algoritmi sunt aceleași. Ambele folosesc aceeași structură daU și aceeași procedură de bază. Ei efectuează un apel de subrutină numai la punctele de contur, astfel încât numărul de astfel de apeluri să fie egal cu aproximativ numărul de intervale de contur (cu alte cuvinte, numărul de noduri din C-LAG). Pixelii din interiorul regiunii sunt abordați într-o manieră secvențială și pentru fiecare dintre ei se face doar un simplu test: dacă sunt sau nu pe contur. Adresarea secvenţială poate fi implementată foarte eficient cu *registrele index* pe majoritatea maşinilor. Pe de altă parte, algoritmul recursiv face aproximativ patru apeluri de subrutine pentru fiecare pixel din interior. Astfel de apeluri sunt costisitoare din cauza necesității de a salva registre

ca informaţie necesară pentru revenirea la punctul iniţial. Mașinile care au o implementare hardware a unei stive pot efectua ultima operație în mod eficient. Prin urmare, costul major al algoritmului recursiv este costul de salvare a locațiilor de memorie în timpul apelurilor subrutine (în cazul în care atât programul apelant, cât și programul apelat folosesc acele registre). Deși un astfel de cost este destul de mare atunci când se folosește un limbaj de nivel înalt, ar putea fi posibil să se conceapă implementări în limbaj de asamblare unde costul este modest (vezi problema 8.5).

Apoi comparăm cei doi algoritmi nerecursivi unul cu celălalt. Umplerea conectivității are două avantaje majore: (a) nu se va scurge atâta timp cât conturul este conectat, (b) timpul de calcul este proporțional cu aria de umplut în loc de aria unui dreptunghi care înscrie conturul, așa cum este cazul algoritmilor de verificare a parității. Dezavantajele sale majore sunt: (a) necesitatea de a cunoaște un punct al interiorului, care în unele aplicații poate fi dificil de găsit, (b) dacă o figură are un contur conectat, dar un interior neconectat, atunci va fi umplută doar partea care conține sământa (un algoritm de verificare a parității va umple toate părțile), (c) interiorul nu este parcurs în ordinea rasterului, ceea ce poate provoca o comunicare excesivă între computer și controler. Acest dezavantaj este relevant în contextul discuției din Secțiunea 8.3.2. Dacă nu putem programa microprocesele sor. apoi putem citi toate liniile orizontale în secvență, păstrând trei dintre ele în memoria principală la un moment dat și evaluăm gradele LAG-ului acolo. Într-o verificare de paritate, odată ce o linie a fost examinată, nu trebuie să fie examinată din nou. Acesta nu este cazul unui algoritm de conectivitate. (Din cauza acestui factor este necesar să se facă umplerea cu o intensitate diferită de cea utilizată pentru contur.) Astfel, costul real de calcul poate fi mai mare din cauza numărului mai mare de operații de intrare și ieșire.

Deoarece ambele tipuri de algoritmi folosesc aceeași procedură de bază, *LINK* $_t$ este posibil să se creeze un algoritm combinat care umple unele părți ale unei imagini printr-o verificare de paritate și altele prin seeding. Acest lucru poate fi important în aplicațiile în care este dificil să determinați o sământă în avans. Poate fi realizat prin rularea algoritmului 8.3 fără intervale de completare și fără ieșire dacă este setat indicatorul de eroare. Când se găsește o linie care provoacă o condiție de eroare, atunci se alege o sământă. Este posibil să relaxați starea de pe grade și să cereți ca acestea să fie mai mari sau egale cu unu. mai degrabă decât egal cu unul. (Această formă relaxată a fost utilizată în implementarea care a produs exemplele prezentate în Figura 8.12 (Planca 26) și Figura 8.13.) Umplerea poate continua apoi prin conectivitate pentru a evita scanarea în afara zonei de contur.

În schimb, putem alege verificarea parității ca algoritm de bază

și apoi revenim la conectivitate când întâlnim linii în care interiorul nu este bine definit pe baza parității. Un astfel de algoritm nu va rata părți din interior care nu sunt conectate la regiunea care conține sămânța.

De asemenea, merită subliniat că algoritmul 8.5 face cea mai mare parte a calculării în jurul extremelor conturului în direcția verticală, puncte care sunt relativ rare. Figura 8.12 (Planca 26) prezintă un exemplu de regiune umplută de algoritmul 8.5 cu un nivel de gri diferit utilizat de fiecare dată când un pixel este luat din absorbție. Algoritmul poate fi adaptat cu ușurință pentru a umple modele, mai degrabă decât uniform, prin selectarea unei culori de umplere în funcție de adresa pixelilor. Un astfel de exemplu este prezentat în Figura 8.13.

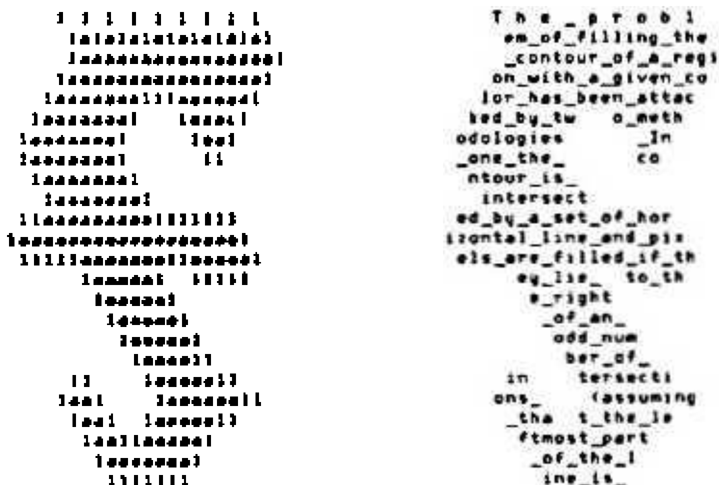


Figure 8.13 *Left:* a contour (marked by l*s) filled uniformly. *Right:* both the contour and the interior have been replaced by text. Spaces were replaced by underlines (_).

20.6 NOTE BIBLIOGRAFICE

Umplerea marginilor este cel mai vechi și probabil cel mai utilizat algoritm de umplere în grafică. Utilizarea sa este clar justificată în aplicațiile în care același contur este afișat în mod repetat, ca și în fotocompunerea. Animația bidimensională în care contururile pot fi supuse translației, dar nu direcționării este o altă aplicație în care umplerea marginilor este adecvată. Algoritmul 8.1 este o variație a algoritmului „YX” descris în 11.NS], modificat pe baza anumitor observații asupra algoritmilor implementate în fotocompozitoare. Efortul major în astfel de algoritmi se îndreaptă spre codificarea corectă a contururilor. Asemenea algoritmi pot fi destul de irositoare în aplicațiile în care un anumit contur va fi umplut de foarte puține ori. Cel mai bine este să folosiți algoritmi bazați pe pixeli acolo, mai ales într-un mediu grafic raster. Trebuie să plătiți un preț pentru eliminarea informațiilor topologice analogice despre contur, deși în anumite aplicații aceste informații nu au fost niciodată disponibile. O serie de lucrări sugerează metode de rezolvare a problemei atunci când conturul este dat ca o colecție

de pixeli fără nicio ordine. [8.DU], [8.ME] și |8.PA) utilizează modificări ale verificării parității. Ultima referință prezintă extensii ale algoritmului 8.3 care umple corect regiunile care nu sunt pline, cu condiția să nu existe pixeli care să conțină mai mult de două arce de contur sau două arce de contur, dintre care unul formează un extremum în direcția y . Algoritmii de conectivitate au fost prezentați în]8.LI], (6.SH] și (8.SM) Algoritmul lui 18.LI] nu este corect și acest lucru a fost subliniat în (6.SH) unde a fost prezentat un algoritm corect, traversând, de fapt, I-LAG.

20.7 LITERATURA RELEVANTA

[8.DU] Dudani, S, A. „Region Extraction Using Boundary Following," CH Chen, cd. *Recunoașterea modelelor și inteligența artificială* **New York; Presa Academică, 1976. p. 216-232.**

(8.L!) Lieberman, H. „Cum să colorezi într-o carte de colorat", *SIG- GRAPH'78.* **Atlanta, Georgia , (august, 1978), pp. 111-116, Publicat de ACM.**

(8.ME) Merrill, RD „Reprezentarea contururilor și a regiunilor pentru o căutare eficientă pe computer". *CACM.* **16 (1973). pp. 69-82.**

(8.PA) Pavlidis. T. „Algoritmi de umplere pentru grafica raster", *CGIP.* **10 (1979), p. 126-141.**

[85M1 Smith. A. R „Tint Fill". *SIGRAF79.* **Chicago. Illinois, (august 1979). p. 276-283. Publicat de ACM.**

20.8 PROBLEME

8.1. Algoritmul de umplere a marginilor din Secțiunea 8.2 nu are prevederi pentru marginile orizontale. Modificați-l astfel încât figurile cu astfel de margini să fie tratate corect.

8.2. Deduceți un format pentru lista de așteptare care stochează doar Ax și Ay plus valorile y la extrema și cât mai puține alte informații posibil.

8.3. Modificați algoritmul 8.3a astfel încât să calculeze valorile lui e și *2 necesare în algoritmul 8.5.

8.4. Estimați costul umplerii unui cerc cu o umplere de margine, o umplere de verificare a parității și o umplere de conectivitate. Repetați pentru cazul unui inel circular.

8.5. (Această problemă presupune că sunteți familiarizat cu limbajul de asamblare.) Elaborați o implementare a algoritmului 8.4 care să nu folosească niciun registru pentru salvarea rezultatelor calculului, astfel încât conținutul acestora să poată fi aruncat atunci când apelați sau reveniți dintr-o subrutină. Apoi apelurile subrutine pot fi aproape la fel de ieftine ca instrucțiunile *de salt* . Comparați acest lucru cu costul implementării algoritmilor nerecursivi.

Capitolul 9

ALGORITMI DE RĂRIERE

9.1 INTRODUCERE

Algoritmii de subțiere au fost studiați pe scară largă în procesarea imaginii și recunoașterea modelelor, deoarece oferă o modalitate de simplificare a formelor picturale. Figura 9.1 ilustrează o motivație pentru un algoritm de subțiere . Pixelii umbriți reprezintă o cuantificare a unui desen de linie care urmează să fie mapat înapoi într-un set de linii. În secțiunile 7.6 și 7.7, am discutat deja cum poate fi definit conceptul de subțire pe o grilă discretă. Vom folosi această analiză aici ca bază pentru algoritmii de subțiere.

Este posibil să se definească subțierea într-un mod riguros din punct de vedere matematic pe planul continuu după cum urmează.

Definiția 9.1: Fie 7? fi o mulțime plană, *B* limita sa și *P* un punct în *R.* Un vecin cel mai apropiat al lui *P* pe *B* este un punct *M* în *B* astfel încât să nu existe alt punct în *B* a cărui distanță de la *P* să fie mai mică decât distanța *PM.* Dacă *P* are mai mult de un vecin cel mai apropiat, atunci se spune că *P* este un punct *scheletic al lui R.* Unirea tuturor punctelor scheletice se numește *scheletul* sau ***axa medială*** a *lui R.* □

Această definiție implică faptul că punctele scheletice sunt centre ale cercurilor conținute în întregime în *R,* cu proprietatea că nu există niciun alt cerc cu același centru și rază mai mare conținută în *R.* Figura 9.2 prezintă câteva exemple de schelete cu unele dintre caracteristicile lor majore. Se poate observa că sunt foarte sensibili la zgomot, deoarece a

o mică perturbare a graniței nu numai că provoacă o perturbare într-o ramură, dar provoacă și crearea de noi ramuri. De fapt, avem următorul rezultat.

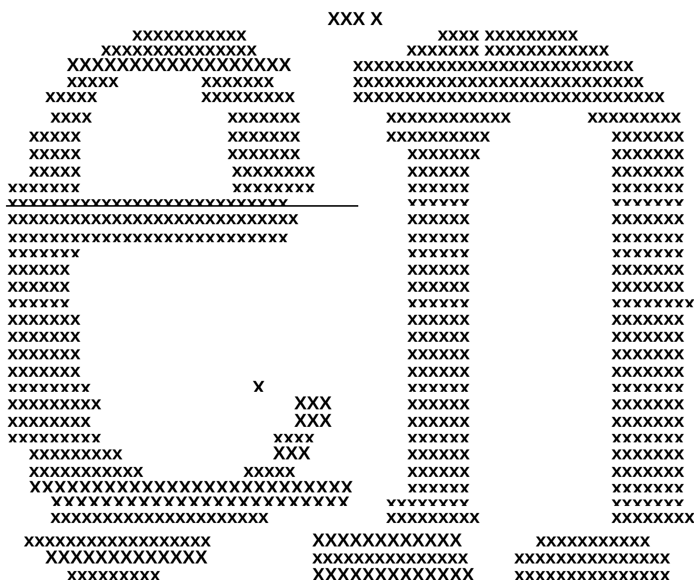


Figura 9.1 Ilustrarea necesității algoritmilor de subțiere: Un desen de linie sau un text trebuie digitizat la o rezoluție suficient de mare pentru a se asigura că nicio linie nu va fi întreruptă și că capetele serif-urilor vor fi păstrate (vezi Capitolul 7). O astfel de rezoluție va determina o lățime mai mare de doi pixeli în alte locații. Subțierea este necesară pentru a recaptura structura liniară a intrării fără a-i distruge conectivitatea.

Propoziția 9.1: Dacă P este centrul de curbură al unui punct al graniței 5 a unei mulțimi plane *R* unde curbura lui *B* are un maxim izolat , atunci există o ramură a scheletului care se termină la *P.*

Dovada: O curbură maximă izolată înseamnă că un cerc tangent la granița în acel punct (Af) are radiouri mai mici decât cercurile tangente la punctele învecinate. În plus, prin definiția curbelor tangente, cercul are mai mult de un punct în comun cu curba și, prin urmare, centrul său (P) face parte din schelet. Orice cerc mai mare va fi tangent în două puncte din vecinătatea lui *M* și, prin urmare, centrul său va fi parte a scheletului, dar mai departe de *P.* □

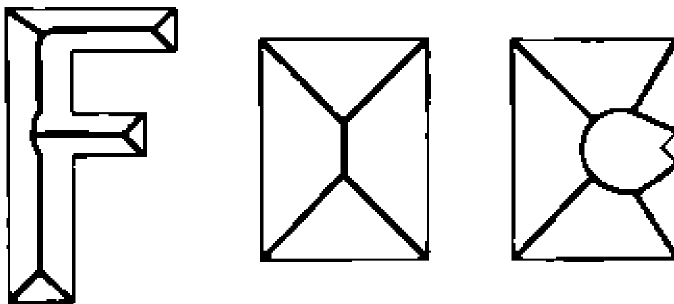


Figure 9.2 **Examples of skeletons: (a) the lineal structure of the silhouette corresponds closely to that of the medial axis; (b) no simple correspondence exists between skeleton branches and object structure; (c) small amounts of noise alter the form of the skeleton drastically.**

O altă observație pe care se poate face din Figura 9.2 este aceea că pentru obiectele care sunt subțiri pentru început, scheletele oferă informații substanțiale despre forma lor. Acesta nu este cazul obiectelor groase, așa cum se arată în Figura 9.2b.

Translația conceptului de axă medială în planul discret nu este deloc evidentă sau chiar fezabilă din cauza dificultății de a defini egalitatea distanței dintre pixeli pe o grilă discretă. Prin urmare, multe sunt lăsate la intuiția implementatorului individual. O alegere posibilă este extinderea Definiției 9.1 la planul discret. S-ar putea defini o versiune discretă a cercului și apoi să caute „cercuri”, conținute în întregime în mulțime și având proprietatea că nu există „cerc” mai mare cu același centru conținut în mulțime. Deoarece o astfel de abordare necesită calcule ample, nu a fost foarte populară. Majoritatea literaturii despre subțiere se ocupă de algoritmi definiți direct pe o grilă discretă. Vom prezenta doi astfel de algoritmi în acest capitol. Ambele se bazează pe conceptul de pixeli multipli (Definiția 7.9 și Teorema 7.3). deși primul folosește doar panoul definiției. Ambii algoritmi modifică un pic conceptul pentru a găsi schelete care au lățimea unui singur pixel, ori de câte ori este posibil. (Reamintim că se pot avea seturi liniare cu lățimea doi, așa cum se arată în Figura 7.17.) Din acest motiv, vom folosi termenul *scheletal* și nu multiplu pentru a descrie abordarea de bază pe care o adoptă ambii algoritmi.

Definiția **9.2:** Scheletul unui set de pixeli /? este un set găsit după cum urmează. În primul rând, pixelii scheletici și pixelii de contur ai lui *R* sunt determinați . Apoi, toți pixelii de contur care nu sunt scheletici sunt îndepărtați și setul astfel găsit îl înlocuiește *pe R.* Procesul se repetă până când rămâne un set format din doar pixeli scheletici. □

Cititorul care este enervat de circularitatea acestei definiții poate înlocui „scheletic” cu „multiplu”. Verificarea dacă un pixel este multiplu necesită examinarea

numai a vecinătății sale imediate și astfel transformările implicate de Definiția 9.2 pot fi implementate prin operații locale (vezi Secțiunea 7.6.2). Partea importantă este presupunerea că se poate decide pentru fiecare set că anumiți pixeli sunt scheletici și să-i păstreze și că alți pixeli cu siguranță nu sunt scheletici , așa că se poate elimina. De asemenea, reamintim că se poate decide fără nicio cunoștință prealabilă dacă un pixel aparține sau nu conturului, deoarece toți astfel de pixeli trebuie să aibă un vecin d cu valoarea zero.

	A	A	A			A	A	A	
	0	P	0			A	P	0	
	B	B	B			A	0	2	

(a)

A	A	C
0	2	2+
B	B	C

(b)

Figura 9.3 Modele de vecinătate ale mai multor pixeli (a) Cel puțin unul din fiecare grup de pixeli marcați cu A sau B trebuie să fie diferit de zero, (b) Cel puțin unul dintre pixelii marcați C trebuie să fie diferit de zero. Dacă ambii pixeli etichetați C sunt diferite de zero, atunci valoarea pixelilor etichetați A și B poate fi orice. În caz contrar, cel puțin unul dintre membrii fiecărei perechi marcați A sau B trebuie să fie diferit de zero.

9.2 ALGORITMI CLASICI DE RĂRIERE

Majoritatea algoritmilor din literatură definesc pixelii scheletici ca fiind doar cei care se potrivesc cu oricare dintre modelele din Figura 7.20 sau din Figura 9.3at, adică ei îndeplinesc condiția (a) din Teorema 7.3. Pentru a păstra pixelii care îndeplinesc celelalte condiții, ei impun o ordine foarte specifică prin care se verifică condiția (a). Rețineți că condiția (a) este de fapt un criteriu de conectivitate și dacă este aplicată secvențial unei imagini, o regiune pur și simplu conectată (adică fără găuri) va fi redusă la un singur pixel, ceea ce, desigur, nu este ceea ce ne dorim.

Această dificultate poate fi depășită prin examinarea pixelilor în paralel și prin extinderea Definiției 9.2 pentru a cere ca dacă un pixel a fost etichetat ca scheletic în timpul unei iterații, acesta nu poate fi șters ulterior. Astfel, dacă *Q* este o matrice orizontală de pixeli, îndepărtarea oricăruia dintre pixelii din mijloc va încălca condiția de conectivitate. Prin urmare, doar cei doi pixeli de capăt ai matricei vor fi eliminați, iar

restul vor fi considerați definitivi. Cu toate acestea, rămâne încă o dificultate, deoarece dacă o matrice constă din perechi de pixeli precum cei prezentati în Figura 7.17, atunci niciunul dintre ei nu este critic pentru conectivitate și toți vor fi șterși. Acest ultim obstacol poate fi ocolit printr-un amestec de procesare paralelă și secvențială. Nu verificăm toți pixelii de contur în același timp, ci doar pe cei al căror vecin /V este zero, unde *N* ia succesiv valorile 0, 2, 4 și 6. Apoi, o matrice de grosimea doi va fi mai întâi subțiată la grosimea unu și unii dintre pixelii săi vor fi astfel păstrați. În astfel de algoritmi este necesar să se marcheze pixelii scheletici într-un mod specific, altfel aceștia ar fi șterși într-o iterație ulterioară atunci când nu sunt necesari pentru conectivitate. Algoritmul 9.1 implementează această metodă. Presupune o intrare pe două niveluri cu pixeli etichetați 0 sau 1. În timpul procesului, pixelii setului care urmează să fie subțiți sunt de asemenea marcați cu 2 sau 3. astfel încât, în timp ce verificăm modelele de vecinătate, trebuie să acceptăm acele valori ca indicând prezența unui pixel.

Operația de potrivire a modelului din pasul 9 se realizează cu ușurință prin formarea unui șir ale cărui valori sunt cele ale celor opt vecini ai *p.* Atunci trebuie să comparăm doar două șiruri. Un exemplu de aplicare a algoritmului este prezentat în Figura 9.4.

t Abrogăm figurile 7.20 și 7.25 ca parte a figurii 9.3 din cauza referințelor frecvente la acestea în acest capitol.

Algorithm 9.1 Classical Thinning Algorithm.

Notație: I **este imaginea de intrare.** *P* **este setul de modele de vecinătăți de pixeli scheletici prezentat în Figura 93a, incluzând, de asemenea, o rotație de 90' a primului model și trei rotații de 90* ale celui de-al doilea model. Flag** *remain,* **când este setat la adevărat, este folosit pentru a indica faptul că pixelii non-skcletali pot rămâne. Flag** *skel* **este setat la adevărat atunci când vecinătatea unui pixel se potrivește cu unul dintre modelele din** *P.* **Un „unu" din model se potrivește cu orice element diferit de zero al vecinătății.**

1. Setați steagul *rămâne* la adevărat.
2. **În timp ce** *restul* este adevărat, faceți pașii 3-12.

   ÎNCEPE.

3. Setați *rămâne* la fals. {Nu s-a făcut nicio modificare.)
4. Pentru *j* " 0,2,4 și 6 faceți pașii 5-12.

   ÎNCEPE.

5. **Pentru toți** pixelii *p* ai **imaginii** *fac* **pașii** 6-10 .

   ÎNCEPE.

6. **Dacă** p este 1 **și dacă** /-vecinul său este 0, **atunci faceți** pașii 7-10.

   ÎNCEPE.

7. Setați flag *skel* la fals.
8. **Pentru** toate modelele *P* **faceți** pasul 9.

   ÎNCEPE.

9. **Dacă** vecinătatea lui *p* se potrivește cu modelul *P,* **apoi** setați *skel* la adevărat și ieșiți din buclă.

   Sfârşit.

10. **Dacă** *simbolul* steagului este adevărat, **atunci** setați p la 2 {scheletal pixel), altfel setați p la 3 {deletable pixel) și, de asemenea, setați *rămâne* la adevărat.

    Sfârşit.

    Sfârşit.

11. **Pentru** toți pixelii *p* din *1* , **faceți** pasul 12.

    ÎNCEPE.

12. Dacăp este 3, atunci setp egal cu 0.

    Sfârşit.

    Sfârşit.

    Sfârşit.

13. **Sfârșitul algoritmului.**

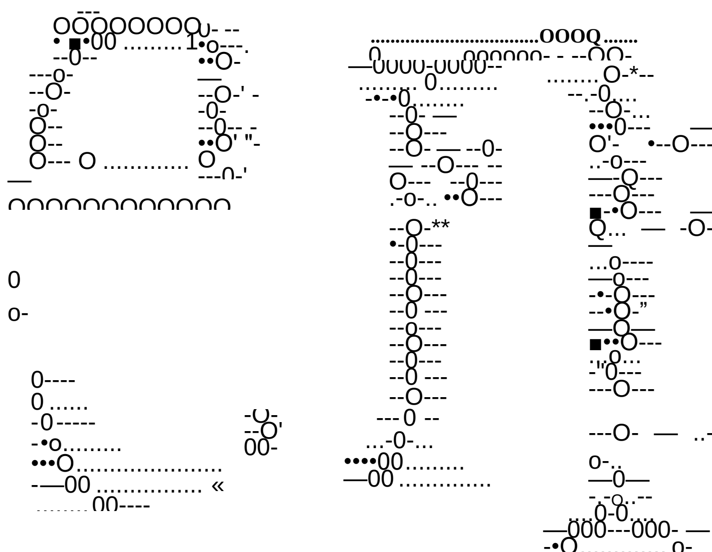


Figura 9.4 **Funcționarea algoritmului clasic de subțiere. Pixelii șterși sunt notați cu „-*, iar pixelii scheletului cu „O”.**

9.3 ALGORITMI DE RĂRIERE ASINCRONĂ

Deși algoritmul clasic specifică o secvență strictă de operații, nu este greu să eliminați această constrângere. Procesarea pur paralelă nu este foarte eficientă pentru imagini, deoarece majoritatea algoritmilor de procesare operează, de fapt, aproape de margini, astfel încât procesoarele alocate pixelilor în interiorul unor zone mari uniforme ar fi inactiv de cele mai multe ori. Pe de altă parte, alocarea unui procesor unei matrice de, să zicem, 32X32 pixeli și procesarea lor secvențială asigură o distribuție mai uniformă a sarcinii de lucru. O astfel de decizie impune următoarele constrângeri asupra naturii algoritmilor care pot fi utilizați:

(1) Deoarece folosim paralelismul, algoritmii care sunt în esență secvenţiali nu sunt acceptabili.

(2) Pentru a evita problemele la joncțiunea domeniului a două procesoare, trebuie să permitem fiecărui procesor să se uite la părți din domeniul vecinilor, dar procesoarelor nu trebuie să li se permită să scrie unul pe domeniul celuilalt. Prin urmare, algoritmului trebuie să i se permită să modifice valoarea pixelului curent, dar nu și valorile vecinilor săi . (Aceasta este o cerință esențială a tuturor algoritmilor paraleli; trebuie să evitați ca două sau mai multe procesoare să încerce să scrie în aceeași locație de memorie. Definiția curentului poate varia în funcție de

algoritm, totuși.) Pentru a menține cantitatea de comunicare între procesoare scăzută, putem insista, de asemenea, că un procesor se uită numai la pixelii din jurul celui actual.

(3) Deoarece o parte a procesării este secvenţială, datele şi algoritmul trebuie organizate în aşa fel încât atunci când pixelii sunt marcaţi, să nu afecteze decizia pentru etapele ulterioare, cel puţin până la o anumită limită când procesoarele pot fi resincronizate. Păstrarea a două copii ale matricei imaginii (una pentru citire, cealaltă pentru scriere) nu este o soluție eficientă.

Orice algoritm care satisface cele trei constrângeri de mai sus are, de asemenea, avantajul că poate fi implementat fie în mod pur secvenţial, fie în mod pur paralel. Se pare că caracterizarea pixelilor multipli conținută în teorema 7.3 poate fi folosită pentru a proiecta un astfel de algoritm .

Singura problemă cu această abordare este că scheletul rezultat necesită editare, deoarece am păstrat ambii pixeli ai căilor cu lățimea doi. Putem obține grosimea unui singur pixel prin stabilirea unei preferințe de orientare și ștergerea unuia din fiecare pereche de pixeli multipli adiacenți, dacă o astfel de ștergere nu va afecta conectivitatea.

Definiția 9.3: Dacă vecinul 4 sau 2 al unui pixel multiplu este zero, atunci pixelul este etichetat ca detașabil, cu condiția ca acesta să nu aibă deja un vecin etichetat ca detașabil. □

Ultima precauție este necesară pentru a preveni obținerea unui obiect deconectat în cazuri similare cu cele prezentate în Figura 9.5.

1	1	2	0	0	0
2	2	d	0	0	0
0	0	4	4	3	3
2	2	eu	d	0	0
1	1	1	2	0	0

Figura 9.5 Model care demonstrează necesitatea utilizării unei etichete suplimentare înainte de a elimina *mai mulți* **pixeli. Pixelii interiori sunt marcați cu 1. pixelii de contur care nu se potrivesc cu niciunul dintre modelele critice sunt marcați cu 2. pixelii de contur care se potrivesc cu unul dintre modelele din figura 9.3a sunt marcați cu 3. iar pixelii de contur care se potrivesc cu modelul din figura 9.3b sunt marcați cu 4 sau d. Ambele imagini etichetate 4 au un 4- sau 2-nvechi care este zero, dar unul dintre ei trebuie păstrat.**

Algoritmul 9.2 Algoritmul de bază de subțiere

Notație: R **este setul de l din imaginea de intrare.** *B(R)* **este limita lui** *R* **și** *M(R)* **mulțimea de pixeli multipli ai lui** *R.*

1. **Repetați** pașii 1-4.

ÎNCEPE.

2. Găsiți mulțimea *B(R)* constând din toți pixelii lui *R* care au un vecin d în afara *lui R.*

3. Identificați setul W(^) al tuturor pixelilor multipli din *B(R).*

4. **Dacă** *B (R)* este egal cu *M(R),* **atunci ieșiți.**

5. Înlocuit *cu R-(B(R)-M (R)).*

Sfârşit.

6. **Sfârșitul algoritmului.**

Algoritmul 9.2 implementează Definiția 9.2. Este independent de modul în care sunt definiți mai mulți pixeli.

9.4 IMPLEMENTAREA UNUI ALGORITM DE RĂRIERE ASINCRONĂ

Cele trei condiții ale teoremei 7.3 pot fi verificate după cum urmează. Definim cinci registre de 8 biți, bdr ^*, bdr* $_2$*, bdr* $_y$ *bdr* $_4$ *și bdr^* ai căror biți sunt setați în funcție de valoarea vecinilor unui pixel. Folosind notația din Figura 7.4, atribuim 0-nvecinul bitului cel mai semnificativ, 1-vecinul la al doilea cel mai semnificativ bit și așa mai departe. Dacă valoarea unui pixel este *J,* atunci biții corespunzători sunt setați în toți octeții *bdr* $_k$ *, k ^ j.* Inițial, toți pixelii au valoarea zero sau unu, astfel încât numai biții din *bdr* $_t$ pot fi setați. Când se constată că un pixel aparține conturului, atunci valoarea acestuia este setată la 2. astfel încât după această operație, biții din *bdr* $_2$ pot fi setați. Putem exprima apariția tiparelor din figura 9.3 într-un mod simplu, ca operații pe biți între aceste registre și măști. Măștile sunt exprimate ca numere octale. De exemplu, pixelii marcați cu *A* în primul model din figura 9.3a corespund cu 160|. Vom folosi notația *a A b* pentru a semnifica un *AND pe biți* între șirurile *a* și *b* , astfel încât *bdr^A* 160, va avea biții care corespund vecinilor 1, 2 și 3 nenuli ai pixelului curent setați la unu. Modelele obținute de rutarea de 90’ pot fi verificate cu ușurință prin deplasarea biților din fiecare registru cu două poziții. Prin urmare, nu ne vom face griji pentru ele în continuare. (Pentru fiecare criteriu stabilit, se găsesc încă unul sau trei prin această operație de schimbare.) Avem următorul rezultat simplu.

Propunerea 9.2: Următoarele afirmații și inversele lor sunt adevărate.

(a) Un pixel aparține conturului dacă *bdr^* 252, nu este egal cu 252,.

(b) Vecinătatea unui pixel are configurația din Figura 9.3a (stânga) dacă Mr ,4 160, # 0 și Mr $_($4 7 * 0 și *bdr /&* 210, este egal cu 0.

(c) Vecinătatea unui pixel are configurația din Figura 9.3a (dreapta) dacă Mr I <4 202, este egal cu 0 și *bdr^* 1 * 0.

(d) Un pixel nu are vecini în interiorul mulțimii dacă *bdr* $_x$ este egal *cu bdr* $_2$ *.*

(e) Un pixel satisface modelul din figura 9.3b dacă *bdr* $_2$ *&* 2OO $_8$ * 0 și *bdr* $_}$ *&* 10, este egal cu 0. și dacă una dintre următoarele condiții este îndeplinită: (el) *bdr* $_t$ *&* 100, și Mr,i 1 sunt ambele mai mari decât 0;

(e2) numai una dintre cantitățile din (cl) este mai mare decât 0 și atât Mr Id 60, cât și *bdr* $_t$ *di 6* sunt mai mari decât 0. □

Dovada: Toate afirmațiile pot fi dovedite prin aplicarea directă a definițiilor. De exemplu, în partea (e), șirul 200, selectează vecinul 0 (cel mai semnificativ bit al modelului este unul). Operația *AND* (d) cu *bdr* $_2$ returnează unul dacă vecinul O este în graniță și zero în caz contrar. Șirul 10, selectează 4-vecinul și operația *AND* cu *bdr* $_x$ returnează unul dacă acel pixel este diferit de zero și zero în caz contrar. Prin urmare, aceste două condiții verifică modelul prezentat de cei trei pixeli din rândul din mijloc al figurii 9.3b. □

Algoritmul 9.3 utilizează aceste rezultate. Pasul 4 localizează pixelii care aparțin graniței, pasul 5 verifică o parte din condiția (b) a teoremei 7.3, iar pasul 6 realizează testul enumerat în Propoziția 9.2(b). Pasul 8 etichetează pixelii care nu au vecini interiori, iar pasul 9 verifică modelul din Figura 9.3b. Pașii 11 și 12 verifică condițiile din Definiția 9.3. La pasul 15 toți pixelii etichetați cu 2 sau 5 sunt șterși. Pașii 6. 9, 11 și 12 trebuie efectuati pentru orientări suplimentare care sunt obținute prin deplasarea modelului de biți cu două poziții.

Algoritmul poate fi implementat printr-un amestec de procesare paralelă și secvențială fără nicio dificultate deoarece operația de etichetare nu afectează valorile registrelor de ordin inferior, adică. valoarea lui Mr, este independentă de valoarea lui *bdr* $_k$ *(k>j* Găsirea contourului și verificarea unora dintre condiții necesită cunoașterea doar a *bdr* $_}$*.* Prin urmare, fiecare procesor poate efectua un set de operații de potrivire a modelelor într-un mod asincron și să aștepte până când toate au ajuns în același punct înainte de a continua.

Algoritmul 9.3 Algoritmul de subțiere asincronă

Notație: C este contorul de iterații plus cinci. *Ștergerea* steagului indică dacă au fost șterși pixeli.

1. Setați contorul *C* la cinci și *ștergerea marcajului* la adevărat.

2. **În timp ce** *ștergerea* este adevărată **, faceți:**

ÎNCEPE.

3. Creșterea *C.*

4. Localizați pixelii de contur conform Propoziției 9.2(a) și pentru fiecare dintre ei **faceți:**

ÎNCEPE.

5. Dacă Mf! are cel mult un bit diferit de zero, **apoi** etichetați pixel cu 3.

6. Dacă sunt îndeplinite condițiile (b) sau (c) din Propunerea 9.2, **apoi** pixelului i se dă eticheta 3.

Sfârșit.

7. **Așteaptă. Pentru** toți pixelii **faceți:**

ÎNCEPE.

8. **Dacă** domnul I este egal cu *bdr* $_2$, **atunci** pixelul este etichetat cu 4.

9. **Dacă** un pixel este pe contur și dacă condiția (e) de
Propunerea 9.2 este valabilă, **apoi** etichetați pixelul cu 4.
Sfârşit.

10. **Așteaptă. Pentru** toți pixelii **faceți:**
ÎNCEPE.

11. **Dacă** *bdr*^ 10, este egal cu 0, **atunci** etichetați pixelul cu 5.

12. **Dacă** *bdr* $_4$ *A* 40, este egal cu 0 și *bdr*^ este zero, **atunci** etichetați pixelul curent cu 5.

13. Dacă un pixel are eticheta 3 sau 4, **atunci** este clasificat ca schele tal și este etichetat cu *C.*
Sfârşit.

14. **Așteaptă. Pentru** toți pixelii **faceți:**
ÎNCEPE.

15. Ștergeți toți pixelii cu etichetele 2 și 5. Dacă nu există pixeli Erased flag *erase* este setat la false.
Sfârşit.
Sfârşit.

16. **Sfârșitul algoritmului.**

Lista algoritmului presupune că oferim fiecărui procesor un segment al imaginii și că fiecare procesor poate citi pixelii adiacente granițelor regiunii sale, dar nu poate scrie pe ei. Prin urmare, nu este nevoie să acordați o grijă deosebită cadrului, cu excepția cazului în care ieșiți în afara regiunii active. Cuvântul cheie wait înseamnă că la acești pași fiecare procesor așteaptă ca ceilalți să termine. In anumite

aplicații poate fi de dorit să se păstreze informațiile despre grosime. Algoritmul poate realiza acest lucru cu ușurință folosind numărul de iterație pentru a marca fiecare pixel atunci când este clasificat ca scheletic pentru prima dată. Acest lucru va necesita utilizarea memoriei suplimentare, dar în rest implementarea sa este foarte simplă. *C* denotă numărul de iterații plus cinci, astfel încât pixelii scheletici pot fi etichetați cu numere care nu interferează cu etichetarea necesară pentru funcționarea principală a algoritmului. O astfel de etichetare poate fi folosită pentru a reconstrui o aproximare a regiunii originale din schelet (vezi problema 9.5). Figura 9.6 prezintă un exemplu de implementare peste caractere alfanumerice digitalizate.

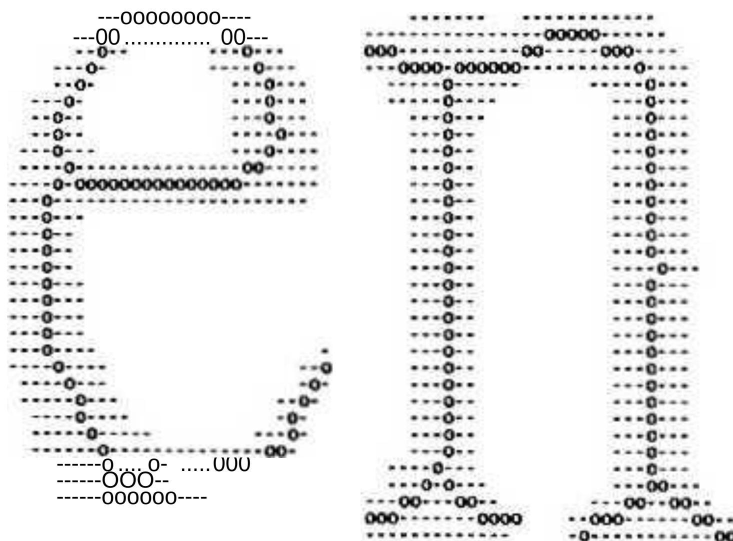


Figure 9.6 Operation of an asynchronous thinning algorithm.
Deleted pixels are marked by '■' and pixels of the skeleton by "o'.

17. UN ALGORITM DE RĂRIERE RAPIDĂ

Algoritmii secțiunilor anterioare efectuează operația de subțiere în mod sistematic pe o zonă a imaginii. Cu toate acestea, există aplicații în care nu este practic să stocați imaginea completă în memorie și în care s-ar dori să se efectueze subțierea într-un mod mai simplu. Algoritmul 9.4 face o formă foarte simplă de subțiere, dar pentru multe obiecte nu va reuși să detecteze părți ale scheletului. Pe de altă parte, este mult mai rapid decât oricare dintre algoritmii din secțiunile anterioare și poate să nu fie o idee rea să-l folosești ca preprocesor, chiar și
cand obiectele nu constau pnmarital din linii subtiri. În acest caz, s-ar putea aplica unul dintre algoritmii generali de subțiere părților siluetei care nu au fost subțiate.

Algoritmul continuă linie cu linie și trebuie să păstreze în memorie doar un număr mic de linii sau părți ale acestora. Pentru simplitate, îl descriem în termeni de parcurgere a graficului de adiacență a liniilor (LAG), dar (graficul de pălărie nu trebuie să fie stocat

în mod explicit. (Vezi capitolul 8 pentru un tratament paralel.) Conceptele de trasee verticale, orizontale și diagonale sunt esențiale pentru funcționarea algoritmului și sunt definite după cum urmează.

Definiția 9.4B: O cale verticală a LAG este aceea **în care** lățimea tuturor nodurilor este sub un anumit prag w_0 și numărul de noduri depășește o valoare dată /V o . □

Definiția 9.4b: O cale orizontală a LAG este aceea în care lățimea tuturor nodurilor depășește un prag NQ-V,, numărul de noduri este sub w_0/v,. iar nodurile de pe fiecare parte a căii au lățimea mai mare sau egală cu w_0. (v, denotă distanța dintre liniile de scanare.) □

Definiția 9.4c: Un traseu diagonal al LAG cu pantă 0 este unul în care toți centrele sunt aproximativ coliniare (acest lucru poate fi verificat prin unul dintre algoritmii din capitolul 12), panta liniei care trece prin ele este 0, lățimea tuturor nodurilor este sub H'osin^, iar numărul lor depășește NQCOS0. □

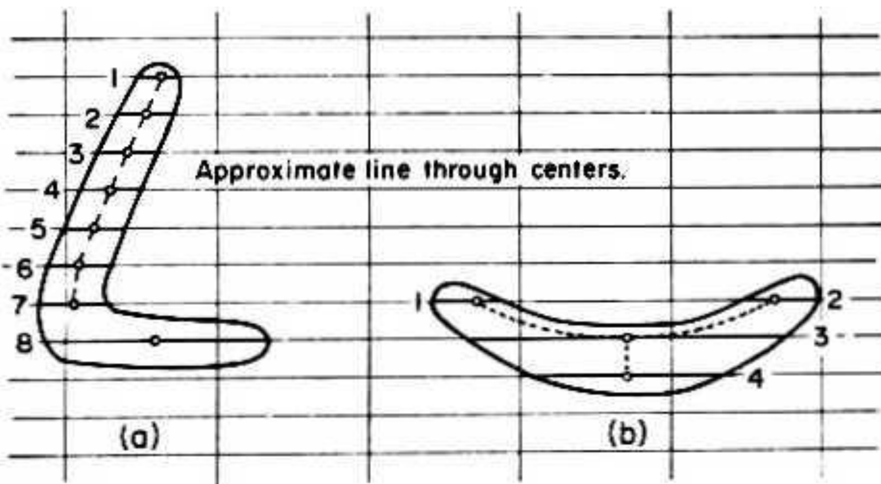


Figura 9.7 **Exemple de siluete în care algoritmul 9.4 funcționează cu succes (a) și fără succes (b).**

Exemplul 9.1: Figura 9.7a prezintă o siluetă și LAG-ul respectiv. Se vede că două ramuri sunt clar identificate și găsite de ritmul Algo 9.4. Deoarece toate lățimile sunt mici, traseul secundar 1-7 este interpretat ca o cursă verticală. Nodul 8 este o cale secundară în sine și este interpretat ca o cursă orizontală. Dacă nodul 7 nu ar fi fost inclus în prima cale secundară, ar fi putut fi folosit ca nod de legătură între cele două linii găsite. În Figura 9.7b, același algoritm nu reușește să găsească cursa curbă. Dacă nodul 3 ar fi fost singur, ar fi fost clasificat ca o cursă orizontală, dar prezența nodului 4 îl descalifică. □

Algoritmul 9.4 Algoritmul de subțiere rapidă

1. Formați LAG-ul și etichetați fiecare nod cu locația centrului și lățimea intervalului

respectiv.

2. Utilizați un algoritm de traversare a graficului (de ex. Algoritmul 6.2) pentru a găsi toate căile formate din noduri cu grad (l,l), posibil începând și/sau terminând cu noduri de grad (1,0) sau (0,1).

3. **În timp ce** oricare dintre condițiile pașilor 5, 6 și 8 s-a dovedit a fi adevărată, faceți:

ÎNCEPE.

4. Pentru fiecare cale găsită la pasul 2 **faceți:**

ÎNCEPE.

5. **dacă** o cale conține o cale secundară care este verticală, **atunci** toate nodurile căii secundare ar trebui să fie etichetate ca aparținând unei astfel de ramuri.

6. Dacă o cale conține o cale secundară care este orizontală, atunci toate nodurile căii secundare ar trebui să fie etichetate ca aparținând unei astfel de ramuri.

7. **Dacă** niciuna dintre condițiile menționate la pașii 5 și 6 nu este adevărat, **atunci fă:**

ÎNCEPE.

8. Căutați o cale secundară diagonală. Dacă se găsește unul iar panta sa este egală cu <£, atunci toate nodurile căii secundare trebuie etichetate corespunzător.

Sfârşit.

Sfârşit.

Sfârşit.

9. **Sfârșitul algoritmului.**

9.6 ANALIZA FORMEI STRUCTURALE

În scurta noastră introducere în analiza formei din Secțiunea 7.8, am menționat că în multe cazuri forma este percepută pe baza structurii generale. Rezultatele algoritmilor de subțiere pot fi folosite ca bază pentru o astfel de analiză structurală și vom revizui această aplicație aici. În general, o analiză structurală trebuie să identifice blocurile elementare ale unui obiect a cărui siluetă este disponibilă ca parte a unei imagini de clasa 2. Acest lucru se poate face direct prin descompunere în forme primitive, simple. De exemplu, s-ar putea exprima silueta ca o unire a unora dintre submulțimile sale convexe. Astfel de metode, deși teoretic posibile, au cerințe de calcul substanțiale. Este posibil să se obțină rezultate similare la un cost mai mic prin analiza scheletelor. Ar trebui să rețineți că, deși această metodă este aplicabilă teoretic oricărui obiect, ea dă rezultate rezonabile doar pentru obiectele care sunt subțiri pentru început.

În practică, analiza formelor bazată pe metode de subțiere poate fi utilizată pentru recunoașterea desenelor în linii sau a caracterelor alfanumerice, obiecte care sunt susceptibile să apară într-un mediu grafic interactiv. Metode suplimentare de descompunere pot fi găsite în literatura citată în Notele Bibliografice. Un obstacol major

în aplicarea tehnicilor structurale a fost lipsa algoritmilor de clasificare convenabil. Tratarea completă a subiectului depășește domeniul de aplicare al acestui text, dar următorul este un rezumat al problemelor majore.

(1) Rezultatele analizei structurale sunt cel mai bine exprimate sub formă de grafic. Este posibil să descriem aceste grafice în termeni de graf gram mars, dar aceasta nu pare a fi o metodă fructuoasă, în ciuda atracției teoretice. Există cel puțin două motive: în primul rând, este foarte dificil să deduceți gramatici din mostre de date și, în al doilea rând, este foarte dificil să analizați gramaticile grafice.

(2) O alternativă la gramaticile grafice este definirea anumitor funcții logice pe graficul (etichetat), de exemplu, „există două linii verticale”. Apoi se poate stabili un arbore de decizie sau se poate defini vectori binari ale căror componente corespund acestor funcții logice. Astfel de vectori pot fi apoi clasificați conform regulilor statistice adecvate. Provocarea aici este selectarea funcțiilor automate ale procedurilor logice.

9.7 TRANSFORMAREA IMAGINILOR BILEVEL ÎN DESENE LINIE

Diminuarea prin ea însăși nu realizează obiectivul de a transforma o imagine de clasa 2 într-o imagine de clasa 3. Scheletele obținute de algoritmii 9.1 și 9.3 sunt încă imagini pe două niveluri, deși cu foarte puțini pixeli negri. Această imagine este echivalentă cu un grafic ale cărui noduri sunt pixelii negri și ale cărui ramuri leagă pixelii cu vecinii lor. Putem transforma acest grafic într-un altul în care nu există noduri de gradul doi, după cum urmează.

Începem o traversare a imaginii pe două niveluri care conține scheletul folosind algoritmul 6.2 sau un algoritm similar. Pe lângă stiva S creăm un buffer *T* unde plasăm coordonatele tuturor punctelor care corespund nodurilor de gradul doi. Dacă un nod are un alt grad decât doi, scoatem coordonatele sale și verificăm dacă bufferul are puncte în el. Dacă da, atunci vom scoate o descriere a acestora împreună cu coordonatele ultimului nod cu alt grad decât doi pe care le-am văzut. Vom explica acest proces prin intermediul unui exemplu.

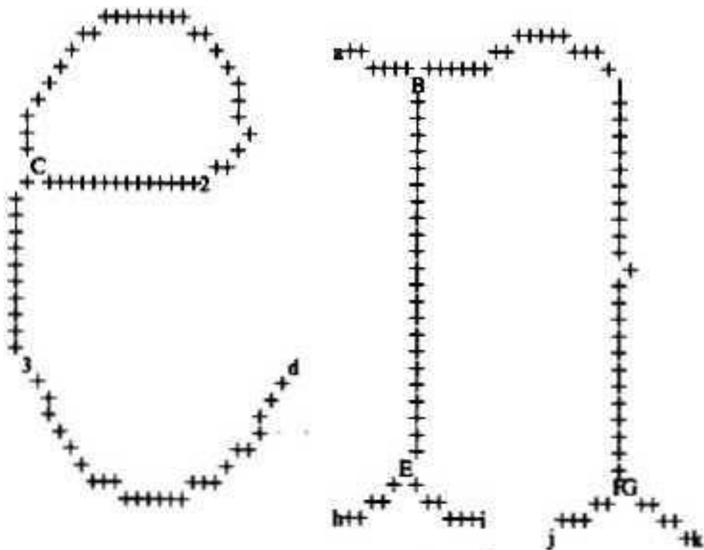


Figura 9.8 Un schelet sub formă de grafic: nodurile de gradul doi sunt marcate cu un „+” sau un număr, nodurile de gradul unu cu o literă minusculă, iar nodurile de gradul trei cu o literă mare.

Figura 9.8 prezintă scheletul exemplului din Figura 9.6, In

pentru a parcurge graficul avem nevoie de un nod de pornire pentru fiecare componentă conectată. Astfel de noduri pot fi găsite de la o scanare de sus în jos, de la stânga la dreapta a imaginii pe două niveluri care conține pixelii scheletici. În general, acesta va fi un nod de gradul doi. așa cum este cazul ambelor componente din figura 9.8. Putem începe să parcurgem graficul fără a produce nicio ieșire până când găsim un nod de alt grad decât doi. Dacă nu poate fi găsit un astfel de nod, atunci graficul este o singură buclă și putem scoate doar acea descriere. Pentru acest exemplu, să presupunem că începem cu nodul „d” pentru prima componentă conectată și cu nodul „a” pentru a doua. Pornind de la „d” aflăm că „C” este următorul nod de grad, altul decât doi. Apoi scoatem perechea „d” și „C” plus o descriere a pixelilor dintre care au fost stocați în memoria tampon *T.* Există o serie de opțiuni pentru o astfel de descriere. Una este să scoată codul în lanț al căii pe care o formează. Aici, va fi

$$5\ ^{5}6545\ ^{2}4\ ^{2}54\ ^{5}34\ ^{2}3*23W$$

unde exponenții denotă simboluri repetate. O altă alegere este să scoateți descrieri ale liniilor și arcelor circulare. Aici, o astfel de descriere va fi raza și centrul unui arc de cerc de la punctul „d” la punctul „3’’, plus o linie dreaptă de la acel punct la „C”. Pentru a obține o astfel de descriere, trebuie să folosiți tehnicile de ajustare a curbei, și în special cele ale aproximărilor pe bucăți descrise în Capitolul 12. Apoi pixelul „2” va fi selectat cu un algoritm de tipul 112.

Din nodul *'C” se va parcurge bucla de sus, care se termină înapoi pe „C”. În acest caz o descriere prin curbe va conține raza și centrul unui arc de la „C” la „2” și

linia dreaptă de la „*2” la „C”.

Pentru a doua componentă, prima ieșire va fi perechea de pixeli „a” și „B”, cu ramura dintre ei descrisă ca o linie. Din „B” putem găsi în continuare „E”, iar din „E”. „h”. Pentru a continua acea traversare trebuie să luăm un nod din suge 5. în acest caz vecinul lui „E” din dreapta jos. Apoi următoarea ramură va fi de la „E” la „i”, și așa mai departe. Rezultatele unei astfel de traversări sunt rezumate în Tabelul 9.1. Secvența exactă a ramurilor depinde de ordinea în care scanăm vecinătatea fiecărui pixel. Aici am ales să privim într-o ordine crescătoare a direcției marcate în Figura 7.4. Am adăugat anumite informații de orientare în descrierea filialelor. Acest lucru poate fi obținut prin tehnicile de ajustare a curbelor descrise în următoarele trei capitole.

Tabelul 9.1: Descrierea figurii 9.8

Puncte finale	Descrierea filialei
d,3	arc în sus
3.C	linie verticală
C.2	arc descendent
2.C	linie orizontală
aB	linie diagonală dreaptă
Fl	linie verticală
Eh	linie diagonală stângă
E ,i	linie diagonală dreaptă
5,1	arc descendent
I. F	linie verticală
F,G	o ramură prea mică
FJ	linie diagonală stângă
Fk	linie diagonală dreaptă

tPoinl (7 este considerat identic cu *F .*

Nu prezentăm un algoritm explicit pentru obținerea unor astfel de descrieri deoarece implementarea exactă a traversării se poate face în mai multe moduri, în funcție de aplicație.

9.8 NOTE BIBLIOGRAFICE

Transformarea axei mediale a fost propusă pentru prima dată în [9.BL]. (9.MO) descrie o soluție pe plan analog prin aproximarea granițelor cu poligoane și apoi rezolvarea sistemelor de ecuații liniare pentru locurile punctelor echidistante. Algoritmul clasic de subțiere este în esență unul propus inițial în (9.SR] și dezvoltat în continuare în (9.TA). Alte lucrări timpurii includ [9.HI], [9.PF], (9.RO1. și (9.RPJ. Legătura dintre modelele specifice de vecinătăți și multiplicitatea traversării este prezentată în (9.YTF). descriși în]9.ACL] și [9.PA]. (Astfel de algoritmi au avantajul că sunt mai rapid de implementat pe computerele secvențiale de uz general obișnuit decât algoritmii paraleli.)

Exemple de descrieri structurale simple pot fi găsite în [9.BE), [6.GR], [9.HTW] și (9.NAM). [6.GR] folosește LAG pentru o procedură de decompunere. Metoda descrisă

în [9.NAM] realizează, de fapt, o analiză a LAG pentru a găsi scheletul. [9.BE] descrie un algoritm de recunoaștere directă a numerelor pentru recunoașterea lui . Este interesant de observat că, deși descrierile structurale au fost

propuse destul de devreme (încă din 1959), au găsit o utilizare limitată în practică. Motivul cel mai probabil este că acestea tind să aibă cerințe de calcul mai extinse decât tehnicile pur euristice. Pe măsură ce prețul calculatoarelor scade, totuși, se așteaptă ca implementarea lor să devină ieftină. Prin urmare, aceștia ar trebui să fie reținuți ca potențiali candidați în orice proiectare de sistem. Vezi (3.PA), I7.FU1], [7.FU21 pentru mai multe despre tehnicile structurale.

9.9 LITERATURA RELEVANTA

[9.ACL1 Arcelli, C.; Cordelia, LP; și Levialdi. S. „De la Maxima locală la scheletele conectate”, *IEEE Trans. Pattern Analysis Machine Intelligence,* **PAMI-3 (1981), pp. 134-143.**

[9.BE) Beun. M. „O metodă flexibilă pentru citirea automată a numerelor scrise de mână”, *Philips Technical Review.* **33 (1973), Partea I: pp. 89-101. Partea a II-a: pp. 130-137.**

19.BL) Blum, H. „O transformare pentru extragerea de noi descrieri ale formei”. *Simpozion despre modele pentru percepția vorbirii și a formei vizuale.* **MIT Press, 1964.**

[9.HI) Hilditch, CJ „Linear Skeletons from Square Cupboards”, *Machine Intelligence,* **4 (1969), pp. 403-420.**

(9.HTW) Hattich, W.; Tropf, M.; și Winkler, G. „Combinație de recunoaștere a modelelor statistice și sintactice - aplicată la clasificarea numerelor scrise de mână neconstrânse,” *Proc. Al patrulea intern. Conf. comună privind recunoașterea modelelor.* **Kyoto, noiembrie 1978, p. 786-788.**

(9.MO) Montanari, U- „Schelete continue din imagini digitalizate”, *JACM.* **16 (1969), p. 5 34-549.**

(9.NAM) Naito, S.; Arakawa, H.; și Masuda, I. „Recunoașterea caracterelor alfanumerice imprimate manual și a simbolurilor bazate pe liniile centroide”, *Proc. Al patrulea intern. Joint Conf on Pattern Recognition,* **Kyoto, noiembrie 1978. pp. 797-801.**

[9.PA) Pavlidis, T. „A Thinning Algorithm for Discrete Binary Images,” *CCIP,* **13 (1980), p. 142-157.**

[9.PF) Pfaltz, JL și Rosenfeld, A. „Computer Representation of Planar Regions by Their Skeletons”, *CACM,* **10 (1972) pp. 119-125.**

[9.RO] Rosenfeld, A. "A Characterization of Parallel Thinning Algo rithms." *Informare și control.* **29 (1975), p. 286-291.**

[9.RPJ Rosenfeld, A. și Pfaltz. JL „Operații secvențiale în procesarea digitală a imaginii”, *JACM.* **13 (1966). p. 471-494.**

[9SR) Stefanelli, R. și Rosenfeld, A. „Some Parallel Thinning Algo rithms for Digital Pictures”, *JACM,* **18 (1971), pp. 2 5 5-264.**

|9.TA) Tamura, H. „A Comparison of Line Thinning Algorithms from Digital Geometry Viewpoint”,

Proc. Al patrulea intern. Joint Conf, on Pattern Recognition, **Kyoto, noiembrie 1978. pp. 715-719.**

19 **.YTF) Yokoi, S.; Toriwaki. J. L; și Fukumura, T. „An Analysis of Topological Properties of Digitized Binary Pictures Using Local Features”,** *CGIP,* **4 (1975), pp. 63-73.**

20 10 PROBLEME

20.1. Scrieți un program care implementează algoritmul 9.1.

20.2. Completați dovada Propoziției 9.2.

20.3. Este posibil să se reducă numărul de etichete utilizate în algoritmul 9.3? În special, putem renunța la utilizarea etichetei *3' pentru pixelii scheletici?

20.4. Scrieți un program care implementează algoritmul 9.3.

20.5. Proiectați un algoritm pentru reconstrucția regiunilor groase din tone de schelet produs de Algoritmul 9.3. Utilizați următoarea regulă: începând cu etichetele mai mari, pentru toți pixelii care au o etichetă dată convertiți toți vecinii lor zero în pixeli cu eticheta cu unul mai puțin și apoi înlocuiți eticheta originală cu 1. Repetați până când toți pixelii care nu sunt zero au eticheta unu. Comparați rezultatele când etichetați numai vecinii direcți cu rezultatele când toți cei opt vecini sunt etichetați. Următorul exemplu arată succesiunea transformărilor - pentru o figură simplă conform regulii vecinătății directe.

```
        »bin              -->            expend
— 1  ................ 3 ........... 6 ....................6
— •11  ............. 22 ............6 ....................66
— 111  ............ 212 ..........7 ..................666
— •1111 ......... 22)2 ............7 ...............666
 .........II ............... 22 ................................6
```

(Reamintim că algoritmul 9.3 adaugă 5 la eticheta de iterație.)

20.6. Dacă sunteți familiarizat cu un limbaj de asamblare, utilizați-l pentru a scrie un program pentru găsirea șirurilor *bdr* $_k$ și pentru verificarea tiparelor de biți - enumerate în Propunerea 9.2 și Algoritmul 9.3. Includeți operația de schimbare pentru a verifica forma modelelor rotite cu 90 de metri.

Capitolul 10

MONTARE CURBA ȘI AFIȘARE CURBĂ

10.1 INTRODUCERE

În Secțiunea 7.6 am discutat cum se poate defini o curbă pe o grilă discretă, iar în Secțiunea 9.7 am discutat cum să derivăm curbe din schelete. Reprezentarea unei curbe ca o secvență de pixeli poate fi adecvată pentru unele aplicații, dar pentru altele am dori să avem o expresie matematică pentru aceasta. O astfel de expresie poate fi mult mai compactă decât formele discrete, după cum arată o comparație între Figura 9.8 și Tabelul 9.1. Găsirea unei curbe care trece printr-o mulțime de puncte date este problema *interpolării,* în timp ce găsirea unei curbe care trece lângă o mulțime de puncte date este problema *aproximării.* Vom folosi termenul de *potrivire de curbă* pentru a ne referi în mod colectiv la ambele. Interesează și problema *afișării curbei : dată fiind o expresie matematică, identificați pixelii care trebuie marcați astfel încât să fie afișată o imagine a curbei descrise de expresie.* Problema nu este deloc banală, chiar și în cazul afișajelor în linie dreaptă (Secțiunea 7.6).

Adesea, cele două probleme apar într-o formă combinată. Setul de puncte care trebuie ajustate printr-o curbă nu este dens, ca în exemplul din figura 9.9, ci rar. Apoi trebuie să conectăm aceste puncte cu curbe și să le afișam pe o grilă în care există mult mai mulți pixeli pe curbă decât în setul original de puncte.

Problemele de potrivire a curbelor apar și în proiectare și producție

ca în grafică, procesarea imaginilor și recunoașterea modelelor. De exemplu, suprafața caroseriei unei mașini poate fi specificată printr-un set de puncte discrete determinate de considerente de inginerie și stil. Pentru a folosi instrumente controlate de calculator, avem nevoie de o descriere matematică a unei suprafețe netede care trece prin toate aceste puncte. Aceasta este una dintre cele mai vechi utilizări majore ale graficii în industrie. O altă aplicație implică reprezentarea datelor experimentale, în scopuri de afișare sau pentru recunoașterea automată a modelelor. În cazul recunoașterii modelelor, descrierea matematică a conturului unui obiect poate oferi informații despre clasa căreia îi aparține obiectul. Cerințele exacte pentru curbe sau suprafețe depind de aplicație, dar rezolvarea tuturor) unor astfel de probleme depinde de o metodologie comună care va face obiectul acestui capitol și al următoarelor trei capitole.

Din punct de vedere matematic, problemele de interpolare sunt probabil mai ușor de rezolvat, dar aproximarea poate fi mai realistă în multe aplicații, în care valorile exacte ale datelor sunt supuse corupției prin zgomot. Un compromis între cele două este de a selecta un set de puncte de ghidare, care pot fi definite interactiv de către utilizator, și apoi de a trece o curbă (sau suprafață) în apropierea acestor puncte. Vom sublinia astfel de tehnici în acest text. Afișarea satisfăcătoare a curbelor necesită rezolvarea unor probleme dificile de geometrie discretă, dar soluțiile *ad-hoc* sunt comune, cu rezultate calitative mixte.

Alegerea formei funcțiilor matematice pentru potrivirea curbei este adesea o întrebare critică. Deși polinoamele vin mai întâi în minte, de obicei sunt alegeri proaste. Cele mai populare tehnici folosesc funcții polinomiale pe bucăți de diferite tipuri. Pentru aproximări, trebuie să ne preocupăm și de alegerea unui criteriu pentru calitatea aproximării. Specificarea unei distanțe maxime a punctelor de la curbă sau suprafață pare o alegere rezonabilă, dar adesea necesită rezolvarea unor probleme de calcul dificile. În general, trebuie să facem un compromis între ceea ce este intuitiv de dorit și ceea ce este fezabil din punct de vedere computațional.

Începem cu potrivirea curbei polinomiale. Ajustarea curbei polinomiale pe bucăți va fi discutată în următoarele două capitole. Accentul va fi pus pe B-spline și aproximări poligonale, în timp ce în acest capitol ne vom concentra pe polinoamele Bezier. Vom acoperi câteva tehnici pentru montarea suprafeței în Capitolul 13.

Afișarea eficientă a curbelor necesită ca punctele afișate să fie generate la un nivel scăzut, de obicei de hardware- ul dispozitivului de afișare. Acest lucru limitează numărul de clase de curbe care pot fi afișate eficient. Cele mai comune două tipuri de arc linii drepte și arce circulare. (Când nu există teamă de confuzie, ne vom referi la ei ca

linii și, respectiv, arce.) Acestea par a fi suficiente pentru multe aplicații deoarece este posibil să se afișeze curbe mai complexe prin ele. Vom discuta problemele lor de afișare în Secțiunea 10.7.

10.2 INTERPOLARE POLINOMIALĂ

Fie (x,,y|),(x $_2$,>' $_2$), ' ' ' (*«>>«) o succesiune de puncte pe planul cu X/ ^ x_y pentru *i* ^ *j*. Se poate obține imediat formula pentru un polinom de interpolare de gradul

(w—1)'*

= JA2^2L_1_!1ZZ1L + (xX|)(xx $_3$) • • - (xx $_n$)

Pn * H Ui-x $_2$) • • • Uj-xJ } 2 (x2-X!)(x $_2$ -x $_3$) • ■ • (x $_2$ -x„)

+ + ' • (*~**-i)

• (x^-X.-j)

sau, într-un mod mai concis.

PM- s^n-^S

/-I *j^i\ * i Xj)*

Putem vedea că *yj* este înmulțit cu o fracție care este egală cu unu când x = x, dar este zero când x ia o altă valoare dintre coordonatele date. Pentru cazul special *n* = 2 găsim ecuația unei drepte care unește două puncte: (10.1)

X—X $_2$ X—X $_1$

+^2 -----

*1”*2 ^2-^1

Un dezavantaj major al interpolării polinomiale este că curba poate oscila semnificativ între punctele date. Un exemplu ilustrează problema. (10.1)

Exemplul 10.1: Următoarele cinci puncte sunt date: (0,0), (1,3), (2,0), (3,0) și (4,0). Polinomul de interpolare este

p(x)--yx(x-2)(x-3)(x-4) (10,3)

P2U) “yi și este prezentat în Figura 10.1a. Are trei extreme, aproximativ la (0,67,3,46), (2,46,-0,47) și (3,5,0,66). S-ar putea să-i fi plăcut ca polinomul să fie aproape de zero în intervalul [2,4] și să aibă un maxim la sau să fie simetric în jurul lui x = 1, dar nu are nicio proprietate. □ (10.2)

Motivul pentru acest comportament de interpolare a polinoamelor este că un polinom este o sumă de puteri ale lui x. Astfel de funcții au proprietatea că valoarea lor pe întreg intervalul este determinată de valoarea lor pe un subinterval arbitrar mic. În timp ce încercăm să manipulăm

coeficienți astfel încât suma să ia valorile dorite în câteva puncte, valoarea termenilor în alte puncte este în afara controlului nostru. Deoarece fiecare termen poate fi destul de mare, nu este surprinzător să existe abateri mari. Astfel, nu există nicio modalitate de a forța o valoare zero pe tot subintervalul [2,4] din Exemplul 10.1. Ar putea fi instructiv în acest moment să privim o interpolare polinomială pe bucăți.

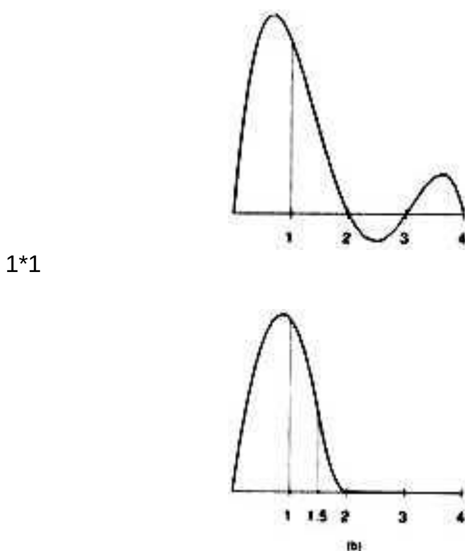


Figura 10.1 Comparația dintre: (a) interpolări polinomiale și (b) interpolări polinomiale pe bucăți

Exemplul 10.2: Aceleași **cinci** puncte sunt interpolate printr-un polinom pătratic pe bucăți. Se introduce un punct intermediar la (1.5,1.35) și se găsește următoarea soluție:

$p_a(x)$	=	6x(0,6-0,7x)	0	<x<	1,5	(10,4a)
PbW	-	$5,4(x-2)^2$	|,5	<x<	2	(10,4b)
P_fU)	-	0	2	<x<	4	(10.4c)

Metodele de selectare a punctului intermediar sunt discutate în Secțiunea 12.4. Aici observăm că aproximarea este continuă și are o derivată întâi continuă, astfel încât diagrama netedă a figurii

10.1 b se obţine. Deși nu există simetrie în jurul x = 1, există simetrie în jurul x = 0,857, iar valoarea funcției este de 3,09. (Simetria exactă ar fi putut fi realizată utilizând patru intervale: [0, 0,5), (0,5, 1,5] etc.) □

Aceste exemple ilustrează faptul că interpolarea polinomială este utilă numai pe intervale relativ mici. Merită atenție doar pentru că este folosit ca bază pentru forme mai adecvate de potrivire a curbei. Cu toate acestea, nu ar trebui să tragem concluzia că interpolarea polinomială pe bucăți este întotdeauna mai bună decât interpolarea polinomială. Dacă domeniul datelor nu este subdivizat corespunzător, atunci orice avantaj al aproximării pe bucăți se poate pierde complet.

Uneori, pe lângă un set de puncte, i se dă și tangenta pe care trebuie să o aibă curba care trece prin acestea la fiecare. Expresia pentru polinomul de interpolare în acest caz este destul de greoaie și este omisă din acest text. Cititorul îl poate găsi în texte de calcul avansat sau analiză numerică. Aici listăm doar formula când sunt date două puncte și tangentele lor: (x^y^ y'i) și (x_2 .y_2 ,y'_2).

ri“?: 1 $(*\sim*2)^2(*``*_I)$

pU) - y'i -

$U|-*2)^2$

y|-?2 $(X-X_I)^2 (X``X_2)$

$U_I-X_2)^2$

X—x2 (10,5)

În general, acesta va fi un polinom cubic. Va fi liniară dacă și numai dacă y'i și y'_2 sunt egale cu panta dreptei dintre (x_u V|) și C^J^)» și pătratică dacă și numai dacă media lui y' și y'_2 este egală cu aceeași pantă.

Exemplul 10.3: Cele două puncte sunt (0,0) și (II) cu pante 0 și respectiv J. Apoi

$$p(x) = U-2)x^3 + (3-s)x^2 \quad (10.6)$$

Figura 10.2 ilustrează două cazuri, pentru 5 = I și J — 16. A doua interpolare este cu siguranță neatrăgătoare. Pe de altă parte, următoarea aproximare pe bucăți produce un rezultat rezonabil pentru s - 16.

$p_a(x)$	-	0	0<x<y	(10.7a)
p^x)	=	$64(x--i)^2$	$V-*-^1$	<' 07b >

Acest lucru este prezentat în Figura 10.2 cu o linie întreruptă. □

Exemplul ilustrează faptul că specificarea tangentelor poate duce la oscilații foarte substanțiale, deoarece un polinom are o derivată constantă de un anumit ordin (aici, în al treilea rând pentru că avem un polinom cubic) și, prin urmare, nu poate face viraje foarte ascuțite.

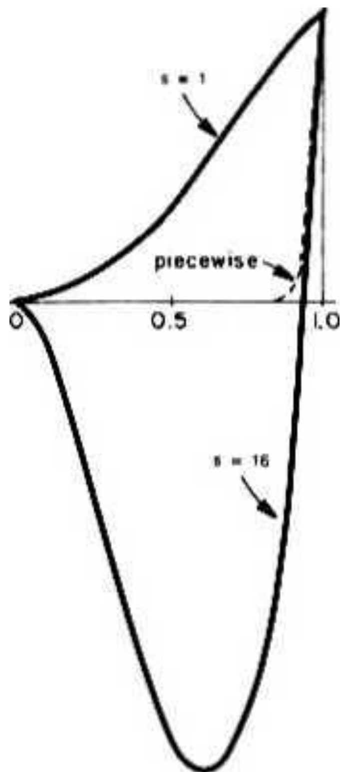


Figura 10.2 Interpolări polinomiale și pe bucăți când sunt date ungenu la punctele finale? Linia întreruptă arată o aproximare pe bucăți pentru s - 16.

10.3 POLINOMIILE BEZIER

Această clasă de polinoame a fost folosită în graficele interactive pentru a obține soluții aproximative la problemele de potrivire a curbelor. În loc să folosiți punctele de date direct pentru a specifica un polinom, un set de puncte de ghidare este specificat interactiv pentru a obține forma dorită. Polinomele au mai degrabă forma parametrică a ecuației (10.8) decât forma explicită *y — p(x).*

x *„PAD*

y-PAD (10.8a)

(10.8b)

Dacă (x 0 ^ 0), (x^)), • • • (x m ,y w) indică punctele de ghidare, polinomul Bezier corespunzător este definit prin

PAD-/\ *cr**' a-o-“'* r-o (10.9a)

MO-s *crt*‘ ^-*tr*-‘*y*, (-0 (10.9b)

unde *C*^ denotă numărul de combinații de *m* obiecte luate *i* la un moment dat și este dat de:

cr m? / f!(m-/)! (10.10)

Această expresie nu este convenabilă pentru calcul, iar *C*^ se calculează cel mai bine din următoarea formulă recursivă:

cr - *cr*^*1* + *cr*_*?* . (10.10)

Este adesea convenabil să scrieți ecuațiile (10.9) într-o formă vectorială prin definirea:

P^*D*- *PAD* / *PAD* and *P,* v / *yi* (10.11)

astfel încât

^(f)- S *cri*‘ (l-r)^ *P*< *.* (10.12)

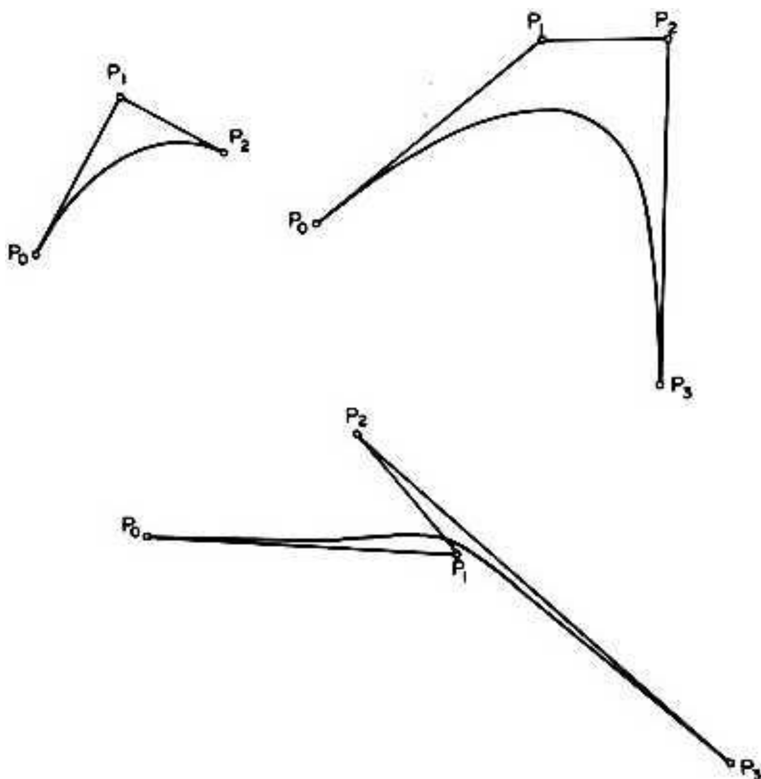


Figura 10.3 **Exemple de polinoame Bezier cu trei și patru puncte de ghidare**

Figura 10.3 prezintă un polinom Bezier pentru m— 2 și două pentru m=3. Ultimul exemplu demonstrează că plasarea punctelor de ghidare pentru a realiza o anumită formă nu este deloc evidentă pentru persoana neexperimentată. Acesta este un defect grav al metodei. Se pare că principalul motiv pentru popularitatea sa este că programele de calculator care implementează polinoamele Bezier sunt mult mai ușor de scris decât programele pentru tehnici mai bune .

Este posibil să se utilizeze ecuația (10.12) nu numai pentru vectorii bidimensionali *P* (așa cum este definit de ecuația (10.11)), ci și pentru vectorii de orice dimensiune. Acest lucru poate fi convenabil pentru descrierea curbelor spațiale. Din ecuația (10.12) este evident că

Ptf»~P $_Q$ și /»(!) = /%, (10.13)

deci intervalul util al parametrului / este de la 0 la 1.

Derivata lui P(/) este

P'(O " "«(l-/T'" P $_o$ +

2 Cfl *ii'~* $^{[}$ (lO" $^{-}$ ' - (™-O/'(l-rr $_{–}^{f-1}$ 1P $_{(}$ + *mt^P^* . (10.14)) ii

Observăm că

P'(0)-m(P, ~P $_0$) și P'(l)-m(P $_w$ -P„-,) , (10.14) astfel încât o expansiune Taylor

aproape de zero produce

P(/) -P(0) + /P'(0) + O(/ ^{2}) = Po(lm/) + *mtP* $_l$ (10.15a) și o expansiune aproape

de unul dă

P(OP(1) -(lr)P'(l) + 0(UO2 >

* Pjl-ma-/)) + m(l-/)P„_, (10.15b)

Astfel, pentru /—-0 și /—»1 polinomul Bezier se află pe liniile care unesc P $_0$ și P| și respectiv P $_m$ _, și respectiv P $_w$, ceea ce înseamnă că aceste drepte sunt tangente ale curbei la P $_0$ și P_m. Mai mult, din moment ce

2 CT"/'(I-/)""' = O+l-rr - 1 /-o

polinomul Bezier se află în interiorul carcasei convexe a punctelor de ghidare.

10.4 CALCULUL POLINOMILOR BEZIER

În continuare, derivăm o relație recursivă pentru polinoamele Bezier care are și o interpretare grafică convenabilă. Rescriem ecuația (10.12) ca

P(/) = (l-/)'"Po + "S c "" (l-')" 1- ' A + ^P« (10.16) /-i

și apoi folosiți ecuația (10.10a) pentru a împărți suma în doi termeni și găsiți asta

F(D“(H) M PO+ 2 cr-'/'a-rf-'P/+ ii

"sc^ /'(i-zy-'p,+/"P„ /-i

sau

P(t) = (i-0«i-')"' , /'o+s' c "~'" ('-')" l "M + /-l

HS'G-7 1 l'-'(H)'-'P l +l"- l ? m |. (10.17)

Observăm că termenii din prima pereche de paranteze formează polinomul Bezier pentru punctele **P $_0$, P $_b$** • • • *P„-u* în timp ce cei din interiorul celei de-a doua perechi formează polinomul pentru *P^, P $_2$, ■' - P„.Dacă* introducem notația *Pu(t)* pentru a desemna polinomul Bezier pentru *P $_{k>}$ P $_k$ + $_h$* ■ ■ *' $_1$*) rew (1) . ca

^(0 - (lz)Po^-i(O + *U') (10.18a)

sau

/WO -PO^-I(0 + HF,.,(I)-*,,..-,(O) (10.18b)

Cu alte cuvinte, un polinom Bezier poate fi găsit din alte două astfel de polinoame prin unirea punctelor corespunzătoare (adică, pentru același z) cu o fină și împărțind acea dreaptă proporțional cu Z. Deoarece această procedură poate fi folosită pentru a obține primele două polinoame, ajungem la algoritmul 10.1.

Algoritm 10.1 Algoritm geometric pentru polinoamele Bezier

Notație: La fiecare iterație R_t sunt punctele de ghidare vechi și *Q,* noile puncte de ghidare.

1. Pentru fiecare valoare a lui *t* de la 0 la 1 de Az faceți:
   ÎNCEPE.
2. **Pentru** *i* de la 0 la m setați $R_t = P_t$.
3. Set $n = m$.
4. **În timp ce** H >0 **face:**
   ÎNCEPE.
5. Pentru i de la 0 la nl setați *Q „** $R_t + t(R_w$ - J?,).
6. Decrementează *n* cu unu.
7. **Pentru** I = 0 la n setați $R_t - Q_t$.
   Sfârşit.
8. P(z)-K 0
   Sfârşit.
9. **Sfârșitul algoritmului.**

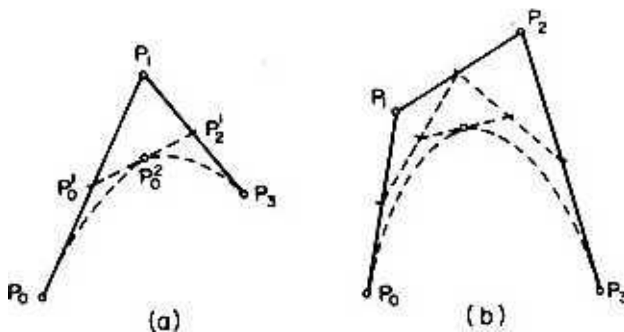


Figure 10.4 **Geometric construction of Bezier polynomials for *t* -0.5. Each new set of guiding points is defined as the midpoints of the sides of the polygon formed by the previous set. When we are down to only one guiding point, then we have a curve point.**

Acest algoritm poate fi implementat și grafic deoarece operația de la pasul 5 este echivalentă cu selectarea unui punct de pe segmentul de linie care unește *Rj și / ?*, + , la o distanță de *Rt* egală cu o fracțiune *t* din lungimea acelui segment de linie. Astfel, dacă *t* este egal cu *m/n,* segmentul este împărțit în *n* părți și noul punct de ghidare se află la sfârșitul acelei *m* ^{lh} astfel de părți după vechiul punct de ghidare. Figura 10.4 ilustrează construcția geometrică pentru w — 3 și w— 4 pentru f „1/2. Deoarece toți vectorii sunt bidimensionali, pasul 5 necesită *6n* operații scalare: o adunare, o scădere și o înmulțire pe componentă. Atunci întregul algoritm necesită

£ (6«) - 3m(m+!> (10.19) nl

operațiuni pentru fiecare valoare a lui t, fără nicio preprocesare sau utilizarea stocării intermediare. Folosind ecuația (10.12) în timp ce păstrați tabele pentru r ^{z} și (1—f) $^{m " /}$, plus având disponibile valorile coeficienților binomi, necesită *m (m* +5) operații scalare.

Un algoritm și mai rapid poate fi găsit prin rescrierea ecuației (10.12) ca

mD-OO" s *cr* 77- *P,* (10.20a) (-0 ^{1*}

sau

$$P(/)\text{-}(l\text{-}/f \left\{ \frac{t}{1-t} \; \overset{m}{S}\, cr \;\; f\text{-}i \left(\frac{t}{1-t} \right)^{i-1} \right. + P_0 \quad (10.20b)$$

Algoritmul 10.2 Algoritmul lui Horner pentru polinoamele Bezier.

1. Pentru fiecare valoare a lui *t* de la 0 la 0,5 prin Ar **faceți:**

ÎNCEPE.

2. Evaluează (l-/)".
3. ^0=^

4. **Pentru** / = I la m **face:**

ÎNCEPE.

5. *Qi* = -^ &-1 + ^-*i P* *m* -<

Sfârşit.

6. F(0=(H)"eM.

Sfârşit.

7. **Pentru** fiecare valoare a lui *t* de la 0,5 la 1 prin Ar **face: Începe.**

8. Evaluați r”.

9. Co " 'o-

10. **Pentru** *i* „ 1 la m **fac:**

ÎNCEPE.

11. *Q^-^Qi-x + crP,*

Sfârşit.

12. *P{D-^Q^*

Sfârşit.

13. **Sfârșitul algoritmului.**

Aflarea valorii lui *P{t}* din termenul dintre paranteze din ecuația (10.20b). presupunând că valorile lui (lt) ^{M} , t/(l—t). iar coeficienții binomiali sunt disponibili, necesită opt operații scalare. Un calcul similar este necesar pentru evaluarea fiecărui polinom din paranteze, astfel încât efortul total să fie 8m+m ■■ 9m, unde al doilea termen din sumă este pentru calculul lui (lf)". Acesta este de fapt algoritmul lui Horner pentru polinoamele Bezier. Pentru valori mici ale lui m (m<5) algoritmul geometric este mai convenabil, dar pentru Horner este mai convenabil, pentru că valoarea lui mare este mai convenabilă, În interesul preciziei , cel mai bine este să folosiți o duală a ecuației (10.20) pentru a calcula valorile pentru r mai mari de 0.5. Apoi, multiplicatorul //(!—;) va fi întotdeauna mai mic decât I.

Este posibil să urmăriți mai îndeaproape punctele de ghidare folosind mai mult de unul în fiecare locație. Figura 10.5 arată efectul pentru exemplul din Figura 10.4a. Se poate observa că multiplicitatea suplimentară a punctului mijlociu afectează rezultatele în mare parte local. O demonstrație analitică a acestei tendințe poate fi făcută pe baza ecuației (10.14). Termenul dintre paranteze este derivata în raport cu *t* a lui

coeficientul lui *P,* în ecuația (10.12). Un calcul simplu arată că derivata este zero pentru z = r, - *i/m,* astfel încât coeficientul lui *P,* este maxim pentru *I " t t .* Se poate verifica, de asemenea, că valoarea coeficientului scade rapid spre zero pentru *t* diferit de r r Prin urmare, pentru orice dat / există doar câteva puncte de ghidare care afectează forma curbei: cele pentru care *intervalul de i t* este mai aproape de intervalul de creștere a intervalului de a crește. valorile lui *t* unde acel punct este dominant în determinarea formei curbei.

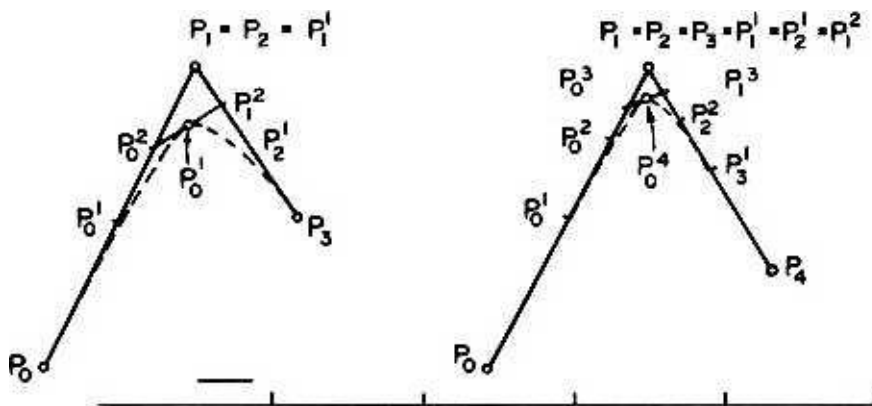


Figure 10.5 Geometric construction of Bezier polynomials with multiple guiding points

Intuitiv, se poate gândi la un polinom Bezier ca la o bandă elastică magnetizată atașată la primul și ultimul punct, cu magneți plasați în alte puncte. Banda este apoi atrasă către fiecare punct și, cu cât este mai puternică intensitatea magnetului acolo (adică, cu cât este mai mare orașul său multiplu), cu atât se va apropia mai mult de acel punct anume. Dacă multiplicitatea tinde spre infinit, atunci polinomul Bezier tinde către arcul poligonal format de punctele de ghidare.

10.5 UNELE PROPRIETĂȚI ALE POLINOMILOR BEZIER

O curbă generată de un polinom Bezier are proprietatea interesantă că orice segment al acesteia poate fi generat și de un polinom Bezier [IO.BE]. Această caracteristică se dovedește a fi utilă în definirea suprafețelor (vezi capitolul 13) și vom afirma și dovedi formal una

caz special. Acolo un nou set de puncte de ghidare este derivat recursiv din cele originale prin înlocuirea unei perechi de puncte cu punctul lor de mijloc. Dacă *k* este numărul iterației, atunci definim

Pf-P, (10.21a)

și avem următoarea relație pentru generarea de puncte de ghidare suplimentare.

Pj-Po *^l (10.21b)

P*-—^# +PA' 1) **l,/^l (10.21c)

Figura (10.6) ilustrează modul în care aceste puncte sunt distribuite.

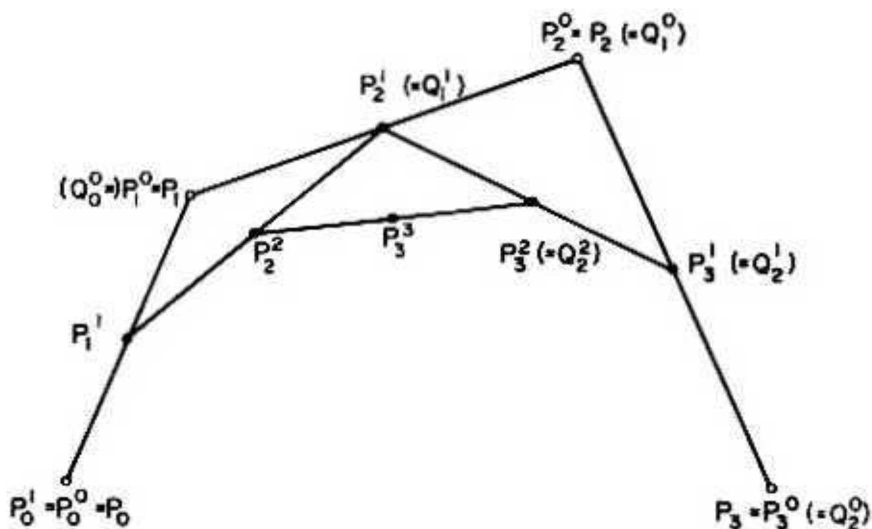


Figura 10.6 **Dispunerea punctelor de ghidare suplimentare pentru un polinom Bezier**

În continuare, formulăm și demonstrăm o teoremă care definește modul în care sunt utilizate punctele de ghidare suplimentare. În acest fel putem găsi o delimitare mai apropiată a formei curbei, așa cum se arată în exemplul din Figura 10.6. Acolo, definiția inițială conține doar două puncte de ghidare în plus față de cele două puncte finale. Ca rezultat al teoremei 10.1, curba poate fi trasată cu cinci puncte de ghidare în plus față de punctele finale.

Teorema 10.1: Pentru 0<f<0,5, valorile lui *P(t}* sunt date de valorile unui polinom Bezier *P* $_a$ *(s)* (0<J<1) cu puncte de ghidare *PQ,* • - • *,Pj.* • ■ • *.P^.Pentru* intervalul 0,5<r<l, valorile polinomului *P(t)* *P^s* arc valorile lui P(t) *P^s}.* (0<s<l) cu puncte de ghidare *PZ* ••• *./',*■'* ••• *.P^*

Demonstrație: Demonstrarea este prin inducție și se bazează pe ecuația (10.18a) repetat aici.

Fo.(D-(l-/)P $_0$ ^-.(0 + r/ $^>$ i«(0.

Această ecuație poate fi rescrisă în următoarele două moduri:

Pftw(O-O“2/)P0^-i(O + 2z **^.(D+^UO** 2 (10.22a)

şi

^(O "2(l-r) 2 + (2/-l)Plw(f). (10.22b)

Dacă setăm m — 1, atunci observăm că PMXO „ Po și Pn(/) „ P|.
Înlocuind din ecuația (10.21), aflăm că pentru m — 1

P(l) - (1-2Z)P? + *21Pj ,* (10.23a)

şi

P(t) - [l-(2/-l)lP|' + (^-DP? . (10.23b)

Înlocuind s - 2/ în ecuația (10.23a), vedem că valorile lui *P(t)* pentru 0<f <0,5 sunt date de partea dreaptă pentru 0<s<l care este atunci un polinom Bezier cu punctele de ghidare *PQ* și *Pj.* În mod similar, substituția J = 2/—l din ecuația (10.23b) arată că partea dreaptă este un polinom Bezier cu punctele de ghidare *Pl* și *Pf* și că, în timp ce s variază de la 0 la l, dă valorile lui P(O pentru / variind de la 0,5 la 1. Prin urmare, teorema este demonstrată sub forma =1.

Pentru pasul general de inducție presupunem că teorema este valabilă pentru m—l. Atunci putem scrie

Po^-iO)» s'p/CT-^dn"'- ^{1} „* * -2/ ro (10.24a)

şi

P1,"(') -ya'C" $^-$ ^^-*^^ (-0 (10.24b)

pentru punctele proprii *Ql.* Se poate verifica cu ușurință că

Ql-PS» C«- ^{2} »

şi prin urmare

||p/+aj-^' **o** ^{026} »

Înlocuind ecuațiile (10.24) în ecuația (10.22a) rezultă

*P^D^PtCr^DD^* ^

/-o

^m^ £ Pl + ^ cm-i^iiq-^Mii (io.27)

i-0 2

Din cauza ecuației (10.26), al doilea termen este egal cu ^m^ 'ZPitïcr-'s , *'(\-s)“- (,+ ". /-o

Acest lucru poate fi scris și ca

^pScr^'s'u-sy"-' .

astfel încât termenii din ecuația (10.27) să poată fi rearanjați pentru a da p.Mo-Po'cr'«-

*)" +

"SP/^"" 1 + C^'p'a-s)" - ' + PmCS»^ COM)

Coeficienții binomi din primul și ultimul termen sunt egali cu unul, în timp ce suma lor din al doilea termen este egală cu C” din cauza ecuației (10.10a). Prin urmare, obținem o expresie identică cu ecuația (10.24a), cu excepția faptului că m— I a fost înlocuit cu *m.* Aceasta completează demonstrația pentru 0<r <0.5. Se lasă ca dovadă pentru exercițiul < r < 10.:·.

10.6 ARCE CIRCULARE

Acum ne îndreptăm atenția asupra anumitor probleme de afișare. Vom discuta întrebările care apar din cauza naturii discrete a datelor în secțiunea următoare. Vă prezentăm aici o problemă specială care apare adesea în aplicațiile grafice: găsirea unui arc de cerc de o rază dată *r* care unește două puncte date, P| cu coordonatele (Xj,yi), iar P_2 cu coordonatele (xid^)- În acest scop trebuie să găsim coordonatele centrului și apoi să selectăm arcul potrivit dintre patru soluții posibile. Începem acest calcul determinând mai întâi coordonatele punctului mijlociu *P„* al dreptei *L* care unește cele două capete.

$$x_m = \frac{x_1 + x_2}{2} \quad (10.29a)$$

yi+y2 ym " 1—

(10.29b)

Lcl jumătățile diferențelor de coordonate să fie notate cu

*i“*2 yi-ya

(10.30)

Centrul se află pe normala la *L* care trece prin punctul său de mijloc. Dacă x i2 este zero (linia £ este verticală), atunci centrul cercului are coordonatele y

^“^ (10.31a)

și următoarele două valori posibile forx (.

x e ^*x m* ± *y/r^-y^* (10.31b)

Dacă £ nu este verticală, atunci trebuie să mai lucrăm. Mai întâi, calculăm panta dreptei £,

u-tan*- — (10.32)

(vezi Figura 10.7) și în al doilea rând, calculăm pătratul distanței *d* dintre centru și punctul de mijloc

^-^-*(xh+yh)* <10,33)

Dacă d^2 negativ, atunci raza dată este prea mică, iar diametrul cercului este mai mic decât distanța dintre puncte. Dacă este zero, centrul coincide cu punctul de mijloc. În caz contrar, avem două seturi posibile de valori pentru centru

""^^T? (1 °.34a)

^^'''^ <l034b '

Expresiile care implică *u* sunt sinusul și cosinusul unghiului ^ prezentat în figura 10.7. (Reamintim identitățile trigonometrice care dă sinusul și cosinusul unui unghi în termeni de tangente.) Ecuațiile (10.34) necesită calcularea mai multor rădăcini pătrate decât este necesar. Dacă înlocuim *d* din ecuația (10.33) și *u* din ecuația (10.32) obținem

X_t=Xm ±y\2V-^—T 1 (10.35a)

y_e -ym +x'A/77y"1 (10.35b)

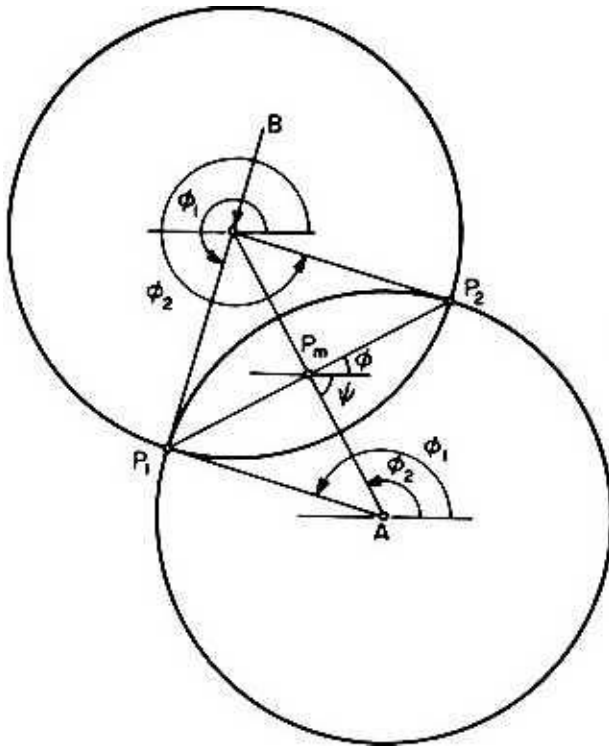


Figure 10.7 **Illustration of the construction for finding a circular arc joining two points with a given radius.** *AB* **is the line segment** *L* **referred to in the text.**

Figura 10.7 ilustrează cele două locații centrale. Pentru fiecare centru avem și două arce posibile, astfel încât, în general, problema unirii a două puncte cu un arc de rază dată are patru soluții. Chiar dacă precizăm o direcție, în sensul acelor de ceasornic sau în sens invers acelor de ceasornic, rămânem totuși cu două soluții. De obicei, cineva dorește să aleagă arcul mai scurt dintre cele două. (Desigur, dacă *d* este zero ambele arce au aceeași lungime.) Dacă definim

X k =x,-x f ^k"^!“^ (10.36a)

*^=*2-^ ^k“^2-K (10.36b) putem calcula unghiul cu axa x format de linia care unește fiecare punct și centru (vezi figura 10.7 pentru notație).

$i=arctanO' k .,x lf) (10.37a)

^2=arctanO' 2f ^ 2 r) (10.37b)

unde funcția arctan returnează un unghi între 0 și 2<. Dacă diferenţa A^ — 0| - 0 7 este pozitiv (cum este în Figura 10.7 pentru locația inferioară a centrului. „A”), atunci un arc în sensul acelor de ceasornic de la primul punct la al doilea punct are lungimea rA^. Dacă diferența este negativă (ca atunci când alegem locația de sus pentru centru în Figura 10.7, „B”), atunci un arc în sens invers acelor de ceasornic are lungimea *—r^tp.* Aceste formule permit selectarea celui mai scurt arc cu o orientare dată.

Uneori avem disponibil un al treilea punct P_3 *și* dorim să selectăm arcul care trece cel mai aproape de acel punct. Dacă (XJ^J) sunt coordonatele sale, atunci putem calcula distanța acesteia de la ambele centre și selectați centrul a cărui distanță este cel mai aproape egală cu raza. Dacă P_3 se află de aceeași parte a dreptei *P\Pi* cu centrul cercului ales, atunci trebuie să selectăm cel mai mare dintre cele două arce, altfel îl selectăm pe cel mai mic. Această decizie poate fi luată prin calculul proiecției segmentului de dreaptă *P^* pe dreapta *L. (*P_c indică centrul cercului: fie „A” fie „B” în Figura 10.7.) Următoarea lemă este ușor de demonstrat printr-un calcul geometric simplu:

Lemina 10.1: Fie *K^* L_y și *M* trei puncte pe plan. Fie $(KL)_X$ diferența cu semne a coordonatei x a lui *K* minus coordonata x a lui *L.* $(KL)_y$, $(ML)_X$, etc. sunt definite în mod similar. Atunci lungimea proiecției segmentului *KL* pe segmentul *ML* este egală cu $\{KL)_X (ML)_X + (KL)_y (ML)_y$ împărțit la lungimea lui *ML.* □

Lema poate fi aplicată direct în cazul nostru luând P_3 pentru *K,* centrul cercului selectat pentru L_y și celălalt centru pentru *M.* Deoarece comparăm proiecția cu jumătate din lungimea distanței dintre centre, împărțirea după lungime poate fi omisă (vezi problema 10.10).

Exemplul 10.4: Dorim să desenăm un arc cu raza 10 de la origine (0,0) până la punctul (5,5), trecând lângă punctul (3,4). Găsim ^ =*y„* =2,5, u - Land

$$\wedge^2 = 100 - \wedge - 87,5$$

sau rf = 9,354. Ecuațiile (10.34) dau apoi următoarele valori

$$x_f ' 2.5 \pm \frac{9.354}{}$$

$$» - 2.5 + \frac{9.354}{VÎ}$$

sau doi candidați pentru centrul arcului: (8.115,—4.115) și (-4.115,8.115). Distanța celui de-al treilea punct față de primul candidat este de 9,59, în timp ce distanța de la al doilea candidat este de 8,76. Primul
valoarea este mai aproape de raza 10. și de aceea selectăm (8.115,-4.115) ca centru. Folosind ecuația (10.36) găsim

x_{lf} - -8.115 $\qquad$ y_k “4.115

x^ - -3.115 $\qquad$ ^ -9.115

iar dintr-un tabel de funcții trigonometrice (sau orice calculator științific) aflăm că 0 (— *2.61rad* și ^ 2 “ 1.90roJ astfel încât lungimea arcului în sensul acelor de ceasornic este (2.67 - 1.90)Xl0 — 7.7. În mod clar, arcul în sensul acelor de ceasornic este mai scurt decât circumferința acelor de ceasornic. Se găsește că lema I este (3-8.H5)(-4.115-8.115)+ (4 + 4.H5)(8.H5+4.115) care este egală cu 12.23 (8.115 - 5.115) sau 36.69. Aceasta este mai mică de două ori J 2. trebuie să fie mai mari trasă în sensul acelor de ceasornic □

10.7 AFIȘAREA LINIILOR ȘI CURBURILOR

În general, o curbă matematică pe plan poate fi dată în una din două forme: reprezentare parametrică

$$x=X(r) \; yr(r) \tag{10,38}$$

sau expresie

$$/(xoO-0 \tag{10,39}$$

Primul este de obicei mai convenabil pentru producerea unui afișaj: *t* este evaluat pentru o serie de valori / b ■■,//, ■■../„ iar pixelii cu
valorile rotunjite de .¥(0 și r(0 ca coordonate sunt afișate. Afișarea reală poate consta din pixeli unici sau linii care le unesc. O alegere critică este numărul de puncte care trebuie afișate. Dacă sunt alese prea depărtate unul de celălalt, atunci curba va avea aspectul unei linii punctate (dacă sunt afișate numai puncte) sau un poligon (dacă punctele nu sunt unite prea aproape împreună cu ele). calcul dar şi într-o curbă care pare prea groasă în părţi Dacă X'(f) şi K'(r) sunt derivate faţă de *t* de

X(t) și *Y(t),* atunci distanța *d* dintre puncte pentru valori ale r Ar unități separate este dată aproximativ de

$$d - M \; VX'(t)^2 + Y\backslash t)^2 \tag{10,40}$$

Astfel, valoarea AZ ar trebui aleasă diferit în diferite părți ale curbei. Din păcate, nu este

întotdeauna ușor de evaluat ecuația (10.40). De exemplu, în cazul polinoamelor Bezier trebuie înlocuiți în el termenii din ecuația (10.14). Când punctele sunt unite prin linii, vrem să evităm unghiurile ascuțite și asta depinde de curbură. Curbura este invers proporțională cu *d* , astfel încât poate fi necesară o strategie de spațiere complet diferită. În mod clar, afișarea unei curbe netede printr-un set de puncte necesită mai mult decât găsirea ecuației matematice a unei astfel de curbe.

Exemplul 10.5: Vrem să afișăm parabola

$$x — r\ y — t^2$$

Calculăm mai întâi distanța dintre o pereche de puncte în funcție de r. Avem:

$$Ax - x_{<+]} - x_{(} - r_{>+i} - 4 - Ar$$

$$Av - y,+i -y, \hat{} t\hat{A}i - r/$$

Atunci distanța *d,* va fi

$$d, - W+Ux)^1 - â» Vl + ('Hl + '.)'$$

1 .c., este o funcție crescătoare a lui r. Rețineți că r, + r $_{/+1}$ este egal cu y'CO ^{a} * un punct intermediar. Dacă vom reprezenta doar punctele, atunci Ar ar trebui să fie ales invers proporțional cu r. □

10.7.1 Afișarea curbelor prin ecuații diferențiale

Forma /(x,^) — 0 poate fi tratată în mod similar cu parametrii dacă putem rezolva în raport cu una dintre variabile. O altă abordare se bazează pe diferențierea ecuației (10.39). Deoarece *f(x^)* este constantă, diferența sa totală va fi zero, astfel încât avem

$$-\&dx +-\hat{}-dy - 0 \quad dx\ dy \qquad (10,41)$$

Introducem notația

$$F \wedge x \wedge = \wedge \qquad 'y\hat{}'\hat{}$$

(10.42)

to obtain

dy —_ ^* (* >>*)
dx F y {xy)

Aceasta este o ecuație diferențială care are curba dorită ca soluție. A fost o metodă obișnuită de a produce afișaje de curbe în dispozitive analogice, deoarece multe astfel de ecuații pot fi simulate prin circuite electronice relativ simple. O aproximare discretă a ecuației poate fi scrisă ca

***Hl'^-** -cfy(XkJk)
n+in- cFAWk) (l 0.43a)

unde *c* este o consUnt arbitrară. Dacă începem de la (x_0 ,y_0). atunci (10.43b) putem găsi punctele rămase din ecuația (10.43). În mod clar, mărimea lui *c* determină densitatea punctelor găsite. Avem aceleași probleme ca și în cazul trasării unei curbe date sub forma ecuației (10.38). Există. totuși, o nouă dificultate semnificativă aici. Deoarece ecuația (10.43) este doar o aproximare a ecuației (10.42), există posibilitatea ca curba rezultată să divergă semnificativ de ceea ce se dorește. O discuție completă a acestui punct implică chestiuni de subilitate numerică și depășește scopul acestui text (vezi Notele bibliografice). Următorul exemplu a ilustrat problema și un remediu într-un caz simplu.

Exemplul 10.6: Ecuația unui cerc de rază *r* cu centrul la origine este

x'+/-P

astfel încât aici ecuațiile (10.43) devin

^+i"x*-2<y i (10.44a)

y*+i *n + 2«* (10.44b)

Dacă dorim să avem y_0 — r și x_0 " 0 atunci putem arăta că ecuațiile diferențelor de mai sus au ca soluție

x* -r(\T+4?)*sin(M)' (10.45a)

*y** -r(\/r+4?)*cos(H) (10,45b)

unde 4 este un unghi a cărui tangentă este egală cu 2c.t Observăm că atât x

t Cititorii care nu sunt familiarizați cu teoria ecuațiilor diferențelor pot înlocui ecuațiile (10.45) în ecuațiile (10 44) și pot verifica dacă ambele »de au aceeași valoare. În acest proces, ar trebui să folosească faptul că expresia \/l + 4c' este egală cu 1/COS0

și y sunt proporționale cu o putere a unui număr mai mare decât unu și anume rădăcina pătrată a lui 1 + 4r^2. Prin urmare, valorile lor vor crește fără limită pe măsură ce *k* crește, iar curba rezultată va fi mai degrabă o spirală exterioară decât un cerc, cu siguranță nu este un rezultat de dorit. Să selectăm acum un număr *b* cu proprietatea care

P + 4c^2 - I

și înlocuiți ecuațiile (10.44) cu

$$x_k \hat{}\sim bx_k - 2cy_k \quad (10.46a)$$

$$\text{mi} \sim prin_k + 2cx_k \quad (10.46b)$$

Selectarea unui astfel de parametru va fi posibilă dacă *c* este mai mic de 1/2. Atunci soluția devine

$$x^* - r\sin(A:0) \quad (10.47a)$$

$$y^* - r\cos(fc\ 0) \quad (10.47a)$$

în timp ce </• este acum un unghi cu tangentă *lc/b.* În mod clar, punctele date de ecuația (10.47) se află pe un cerc, iar arcul dintre punctele succesive este egal cu r0. □

Selectarea valorii lui c trebuie făcută astfel încât punctele să fie suficient de^dozate. În exemplul anterior factorul critic este raportul 2r/Vl —4c*

10.7.2 Efectul erorilor Rouad-off io Afișări

Selectarea pasului de eșantionare adecvat al curbei nu este singura problemă în producerea unui afișaj bun. Structura detaliată a afișării curbei este, de asemenea, afectată de eroarea de rotunjire a valorilor X(/) și HO- Într-adevăr, toate afișajele au o rezoluție finită și trebuie să selecteze pixelul cu coordonatele cel mai apropiat de valorile calculate și să afișeze acel punct, mai degrabă decât unul ale cărui coordonate sunt exact *X(i),Y{t)* (orx*^* dacă metoda din subsecțiunea anterioară a fost discutată în secțiunea anterioară). 7.6 privind definirea liniilor drepte pe o grilă. Să presupunem că vrem să afișăm linia dreaptă de la un punct (Xj.y,) la altul, (XJ^J). Punctele liniei sunt definite de ecuația (10.2). și putem folosi acea ecuație pentru a găsi valorile lui y pentru diferite valori ale lui x.

Exemplul 10.7: Fie x, - y, - 0 și x_2 - 40, y_2 = 17. Atunci ecuația (10.2) devine

Dacă rotunjim la cel mai apropiat număr întreg, obținem următorul tabel

Tabelul 10.1: Comparația coordonatelor exacte și rotunjite

X	1	2	3	4	5	6	7 • • •
exact y	0425	0 85	eu 275	1.7	2.125	2,55	2.975 • • •
y rotunjit	0	1	1	2	2	3	3 ' ' •

care produce o cale /-conectată. Dacă panta este mai mare decât unu, atunci ar trebui să folosim ca variabilă independentă. □

Acest exemplu arată că linia rezultată din trunchiul *y* are un aspect de scară care poate fi neatrăgător din punct de vedere estetic. Am menționat în Secțiunea 7.6 că, dacă C este un punct pe linia dintre J și *B,* linia afișată între .4 și *C* poate să nu facă parte din dreapta *AB,* așa cum este demonstrat în Exemplul 7.1. Dacă avem un dispozitiv de afișare cu scară de gri, atunci putem alege să nu afișam toate punctele la aceeași luminozitate. În punctele în care valorile exacte și trunchiate sunt apropiate, de exemplu pentru x = 2. 5. 7 etc., un singur pixel este afișat la luminozitate maximă. Pentru alte puncte sunt afișați doi pixeli la o luminozitate redusă. Apoi linia rezultată pare mai netedă. Aceeași tehnică poate fi utilizată pentru afișarea curbelor arbitrare. O discuție completă a unor astfel de metode depășește scopul acestui text (a se vedea Note bibliografice). De asemenea, cititorul ar trebui să țină cont de faptul că există multe aplicații, cum ar fi fotocompunerea, unde afișajul trebuie să fie pe două niveluri.

Alte dificultăți pot apărea atunci când calculele de afișare sunt efectuate în procesoare de afișare. Deoarece aceste procesoare au, de obicei, doar aritmetică cu numere întregi de precizie scăzută, coeficienții matematici ai unei curbe ar putea trebui să fie rotunjiți atât de sever încât curbele pot să nu treacă prin puncte despre care se știe că se află pe ei.

Exemplul 10.8: Vrem să unim punctul (0,0) cu punctul (17,11) cu un arc de cerc al cărui centru este pe axa x. Ecuația cercului are forma

$$x^2 - 2x_f x + y^2 - 0 .$$

şi aflăm cu uşurinţă că $2x_f$ - 410/17 - 24.1176 • • • . Acesta este rotunjit la 24, astfel încât trebuie să folosim următoarea ecuație pentru reprezentare

$$F(xy) -x^2 - 24x +y^2 -0$$

Găsim că F(17,ll) — 2 astfel încât punctul final nu aparține curbei. □

A avea punctul final în afara curbei poate să nu fie prea rău dacă

PLACIURI

PLATES

CUPRINS



Figure 2.3 Digital image with 256X256 samples and 256 gray levels



Figura 2.4 Imagine l»ipit.il cu mostre de 64x64, reconstruită pe un displax mare



Figura 2.5 Imagine digitală cu mostre de 32x32, reconstruită pe un afișaj mare



Figura 2.1 O copie a imaginii? 3 cuantificate în X niveluri (trei biți)



Figura 2.10 O copie a figurii 2.3 cuantificată la patru niveluri (doi biți)



Irgure 2.11 I hi he red ^crsnm ul | «Gute 3 9



figura 2.12 Versiunea dithered a figurii 2.10



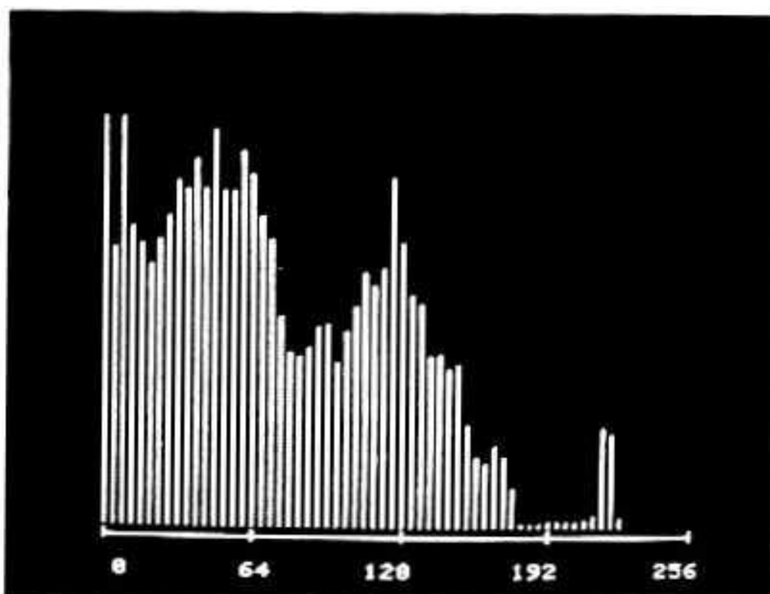
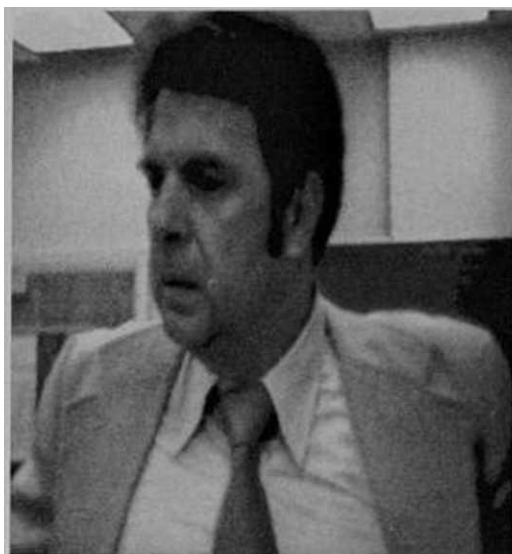
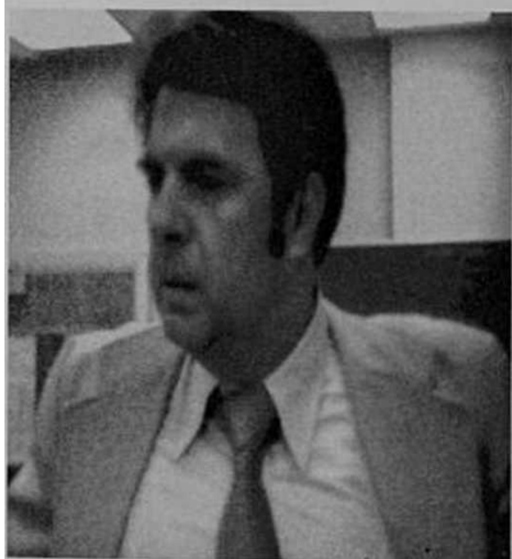


Figura 3.3 Imaginea Au și histograma corespunzătoare care arată că nu toate nivelurile de luminozitate sunt utilizate eficient

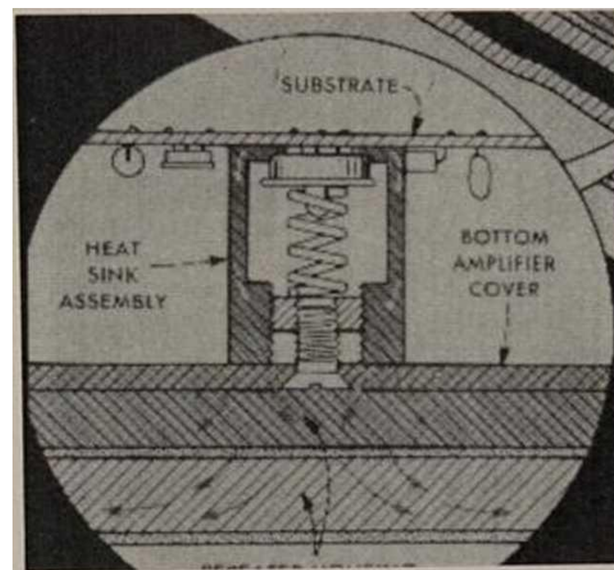


(a)



(b)

Figura 3.4 Rezultatele egalizării histogramei pentru originalul din Figura 3.3. Imaginea de sus a fost produsă de Regula I și cea de jos de Regula 2.



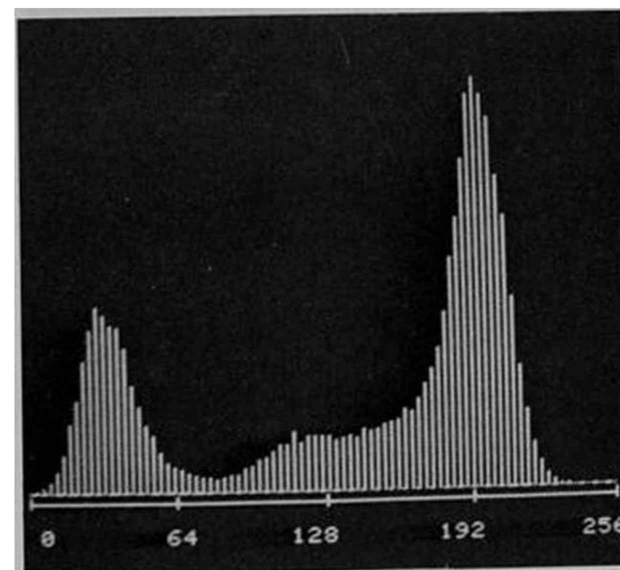


Figura 3.5 O imagine și histograma ei

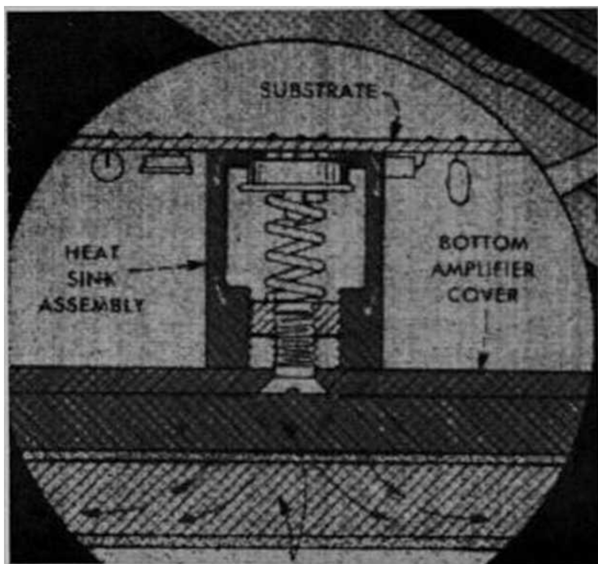


Figura 3.6 Deteriorarea aspectului unei imagini din cauza egalizării histogramei



(a)



(b)

Figura 3.7 Imagini cuantificate într-un mod optim prin egalizarea histogramei: (a) cu patru niveluri și (b) cu opt niveluri. Aspectul este net superior celui obținut cu cuantizarea uniformă prezentată în figurile 2.9 și 2.10.



Figure 3.9 Original of the image used for illustrating the effects of filtering in Figure 3.10



(a)



(b)

Figura 3.10 (a) O imagine zgomotoasă și (b) efectul filtrului liniar asupra acesteia



(o)



(b)

Figura 3.11 Efectele unui filtru direcțional trece-jos asupra originalului din Figura 3.10a: (a) o trecere, (b) trei treceri.

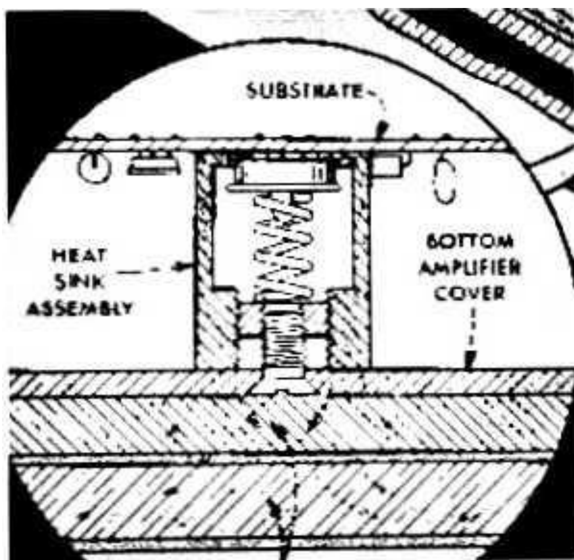
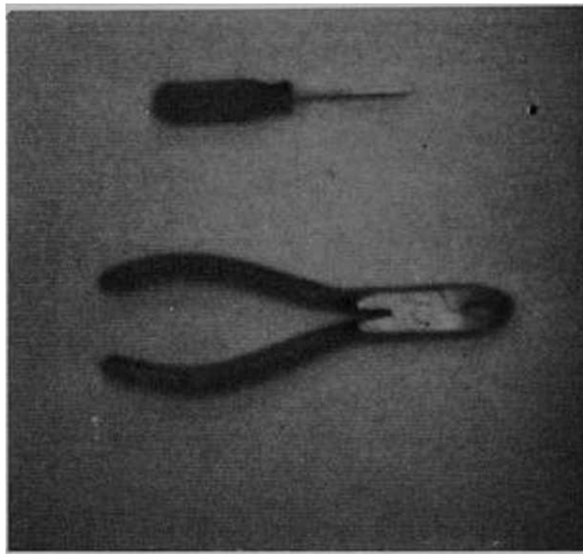
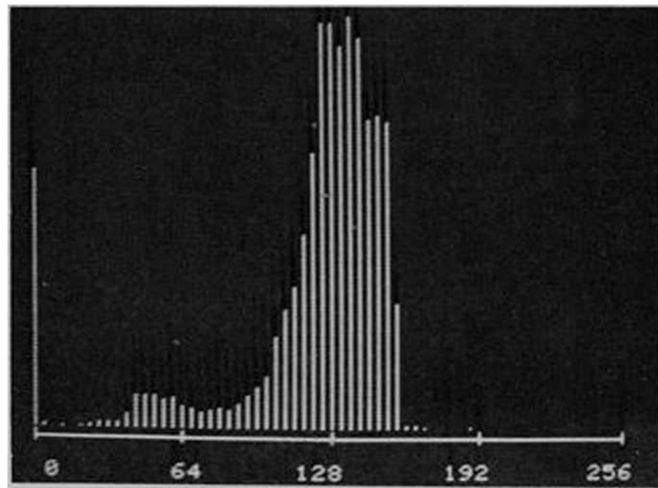


Figura 4.1 Imaginea Thrcsholdcd pentru 7 - 100 din originalul din figura J.5



(a)



(b)

Figura 4.2 (a) Imagine originală (256X256 pixeli); (b) histogramă la nivel de gri.

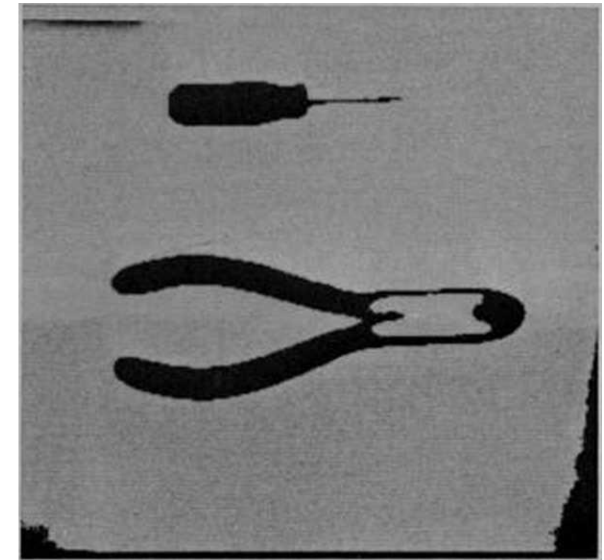


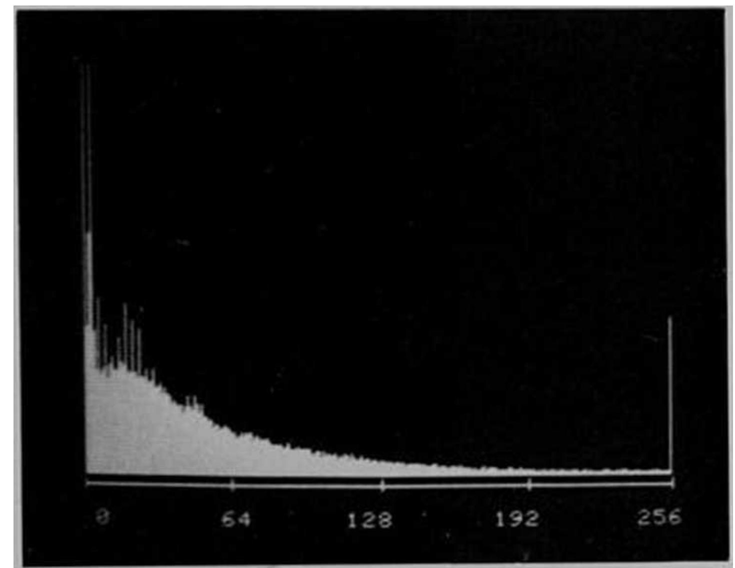
Figure 4.3 Results of thresholding for T = 100 for the original of Figure 4.2a



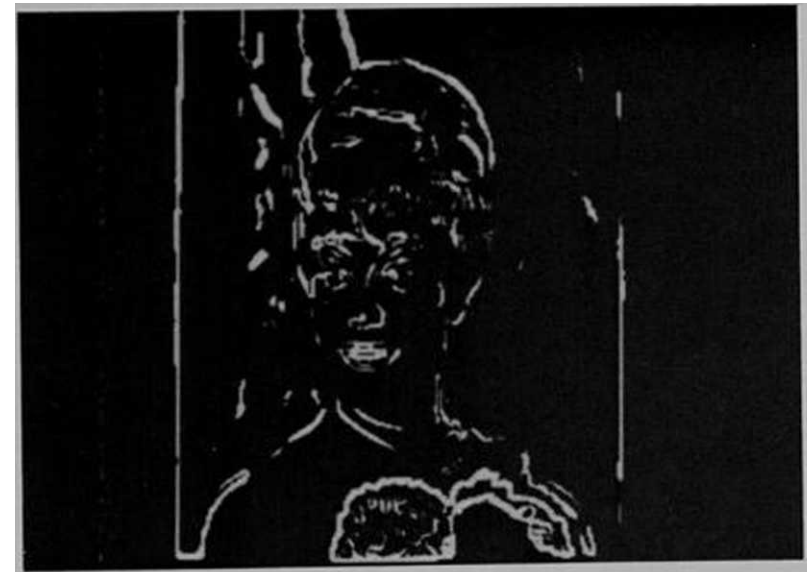
(o)



(b)



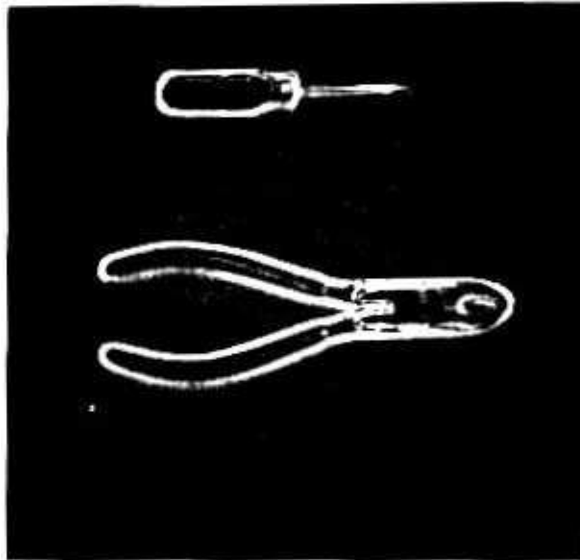
(o)

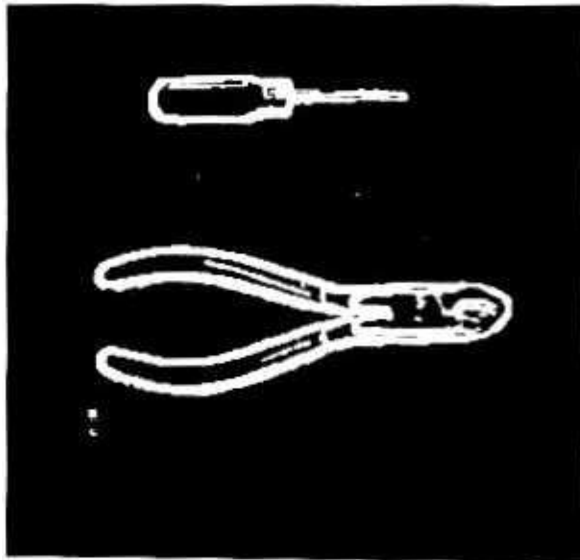


(b)

Figura 4.4 (a) Imagine cu gradient orizontal, (b) margini găsite atât din gradientul vertical, cât și din gradientul orizontal.

Figura 4.5 (a) Histograma din Figura 4.4b. (b) forma thrcsholdcd din figura 4.4b pentru *T* = 128.





(b)

Figura 4.6 (a) Muchii găsite atât din gradienții verticali, cât și pe orizontale pentru originalul din Figura 4,2a. (b) forma thrcsholdcd a (a) pentru T „ 64.

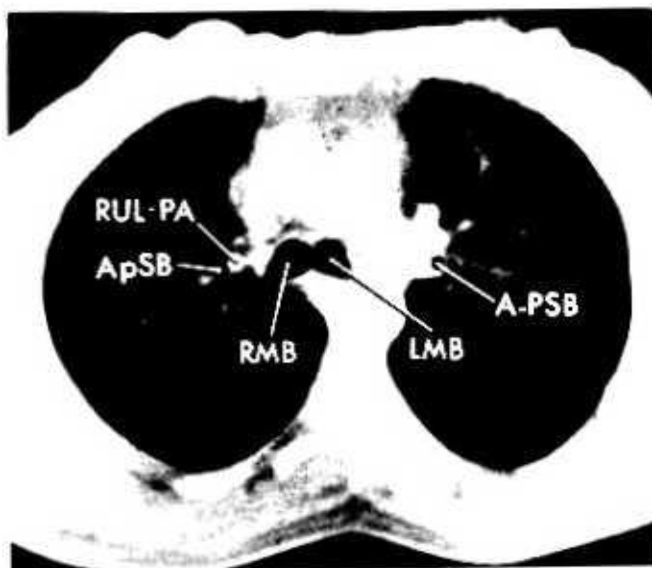
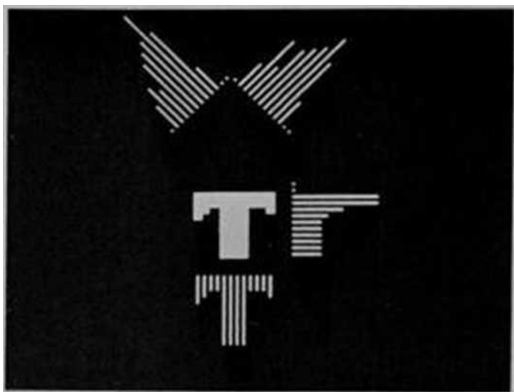


Figura 5.1 **Reconstrucția unei secțiuni transversale a unui corp uman la nivelul superior al pieptului, prezentând detaliile bronhiilor. Eticheta RUL-PA indică» către o arteră pulmonară. Celelalte etichete identifică bronhiile. (De la Naidich, DP; Terry. PB; Slilik. FP; și Siegelman. SS „Computed Tomography of the Bronchi: I. Normal Anatomy"** *Journal of Computer Assisted Tomography.* **4 (1980), pp. 746-753.)** *(Cu amabilitatea SS Siegelman Copyright © 1980 Raven Press)*



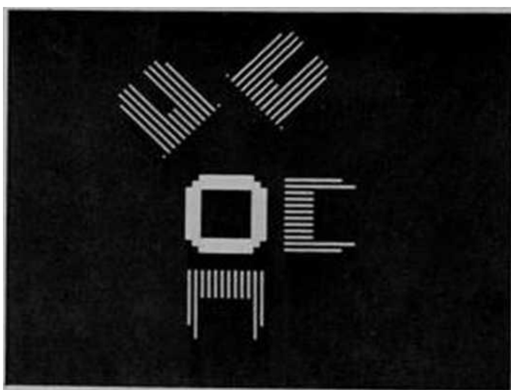
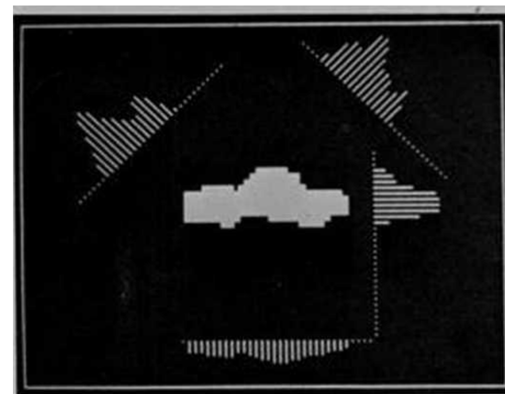


Figura 5.7 Matrici binare reprezentând caractere tipărite înconjurate de proiecțiile respective găsite de algoritmul 5.2



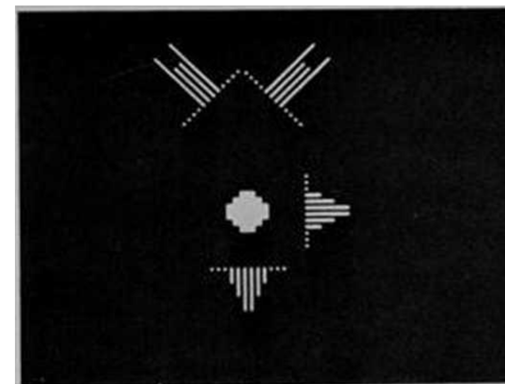
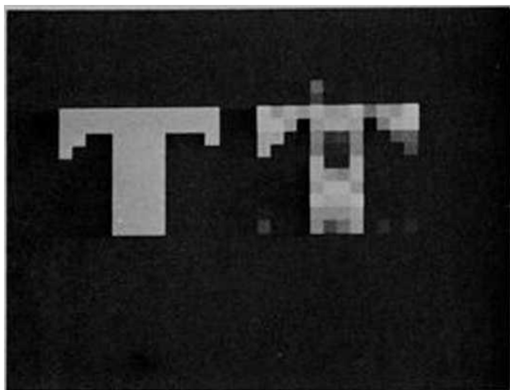


Figura 5.8 Matrici binare reprezentând conturul unei mașini și un disc circular înconjurat de proiecțiile respective găsite de algoritmul 5.2



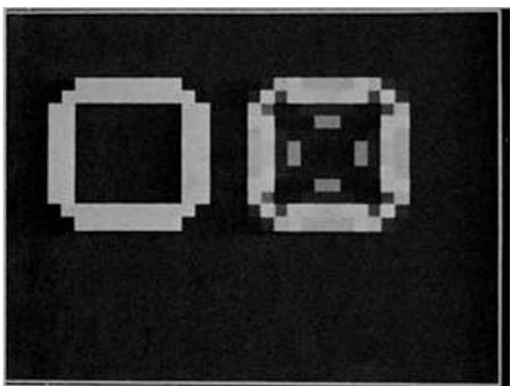
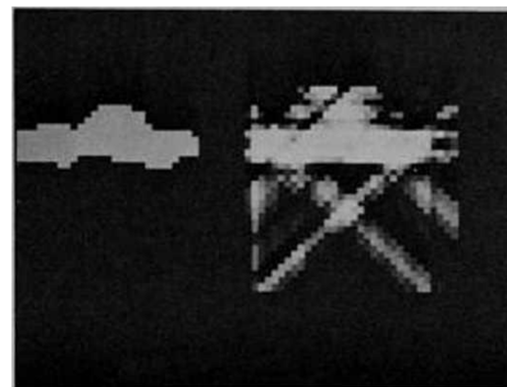


Figura 5.9 Reconstituirea exemplelor din Figura 5.7 folosind retroproiecții filtrate



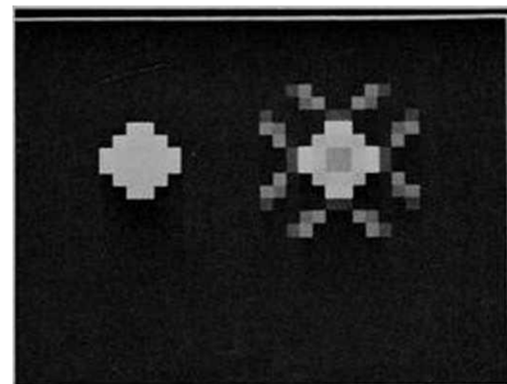


Figura 5.10 Reconstrucție a exemplelor din Figura 5.8 folosind retroproiecții filtrate

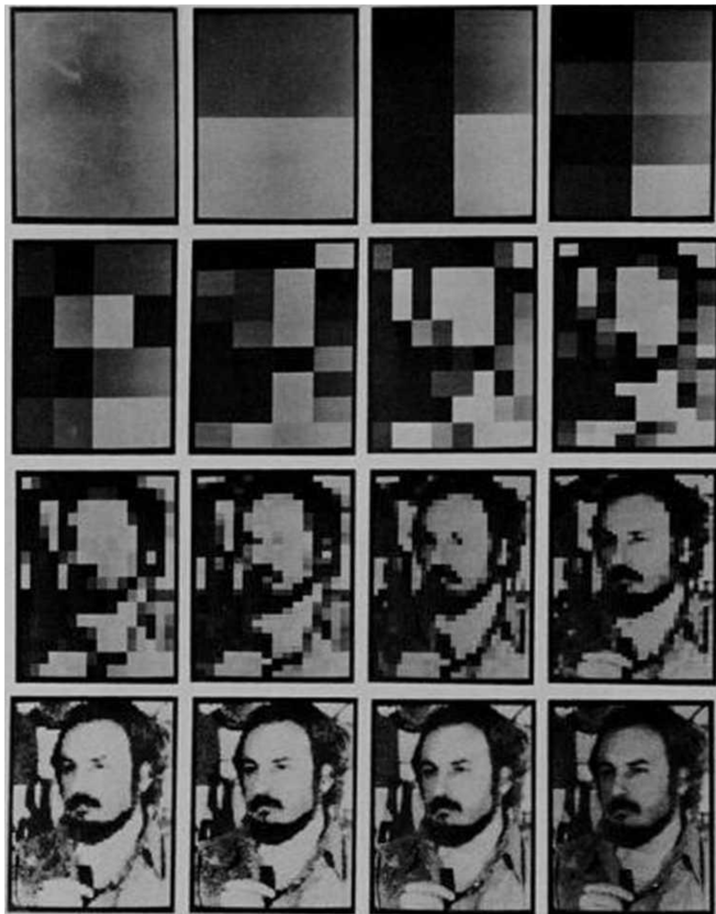


Figura 6.8 Secvența imaginilor produse prin transmiterea succesivă a elementelor arborelui binar. (Din Knowlton, K. „Progressive Transmission of Grey Seale and B/W Pictures by Simple, Efficient and Lossless Encoding Schemes,” *IEEE Proceedings.* **68 (1980), pp. 885-896.)**

(Cu amabilitatea lui K. Knowlton. Copyright © 1980 IEEE}



Figura 6.9 Utilizarea unui arbore quad într-un algoritm de împărțire și îmbinare. Criteriul de uniformitate sa bazat pe o evaluare a matricei de co-ocurență. Limitele pătratelor sunt suprapuse pe imaginea originală. Inițial, imaginea a fost împărțită în 64 de pătrate. Unele dintre ele au fost comasate, în timp ce altele au fost divizate.

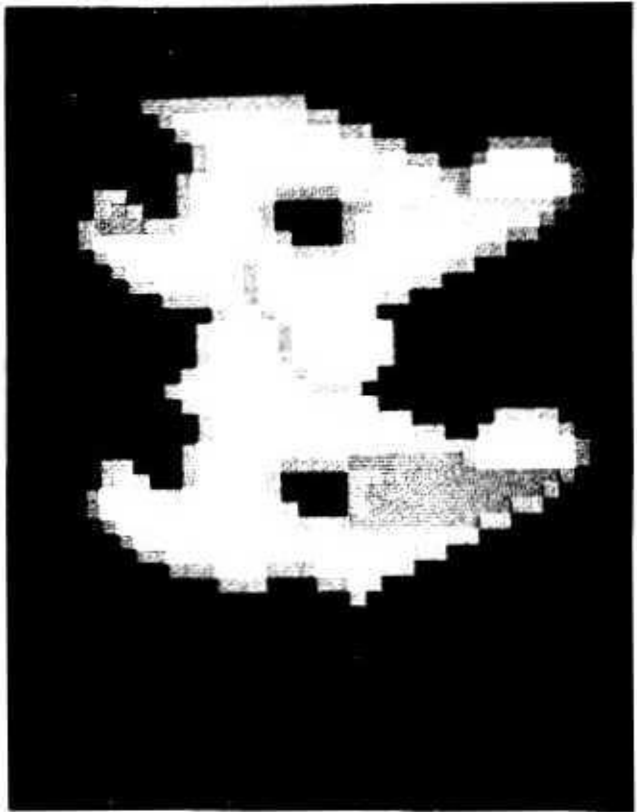


Figura 8.12 Secvența în care sunt umplute diferite vase din interior. Preprocesorul de verificare a parității găsește o sămânță în partea de sus din stânga a interiorului. Umplerea se realizează pe baza conectivității. Fiecare nivel de gri corespunde unui ieșire din stivă.

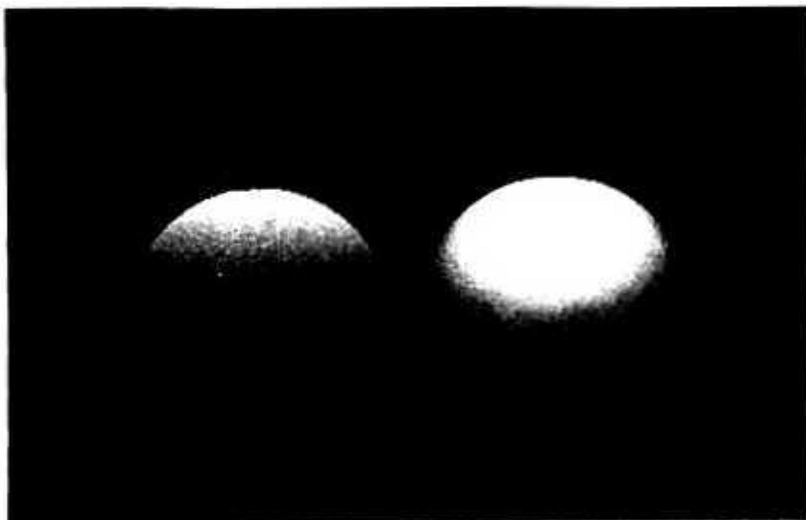


Figura 13-4 Umbrirea unei sfere folosind tehnica din Exemplul 13.7. 1 efl. o sferă mată *{g/d* **- 1/10) și dreapta, o sferă lucioasă*** *lg'd* **" 10/1).**



Figura 17.1 Utilizarea umbririi și a reflexiei produce o scenă realistă dintr-o descriere constând din 36 de pete bicubice în plus față de un număr de poligoane. (Din „A 3-Dimcnsional Representation for fast Rendering of Complex Scenes" de S. M. Rubin și T. Whined, în *SIGGRAPH'80,* **pp. 110-116.)** *(Cu amabilitatea lui Turner Whitted. Copyright © 1980 ACM)*

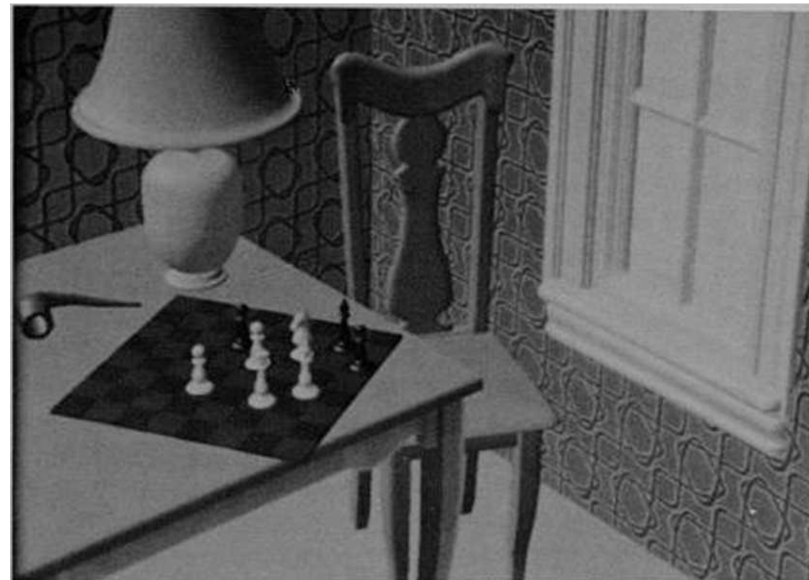


Figura 17.2 Utilizarea umbririi, eliminării suprafețelor ascunse și proiecției în perspectivă creează o scenă care este greu de distins de o pictură. Obiectele scenei au fost descrise de 32.000 de poligoane. (Din „A Software Test-Bed for the Development of 3-D raster Graphics Systems" de T. Whitted și DM Weimer, în *SIGGRAPH 81,* **pp. 271-277.)**
(Cu amabilitatea lui David M. Weimer. Copyright© 1981 ACM)

punctul este inclus în pixelii afișați, dar aceasta depinde în întregime de regula precisă utilizată pentru trasare (vezi problema 10.9). Prin urmare, nu ar trebui să folosiți o verificare a egalității pentru a decide când să opriți trasarea unei curbe, cu excepția cazului în care o analiză atentă arată că acest lucru nu va cauza probleme. Criteriile alternative de terminare verifică egalitatea numai a uneia dintre coordonate și apoi testează dacă cealaltă diferă cu mai puțin de una. (Din păcate, anumite implementări comerciale ale rutinelor de desenare conică folosesc verificări de egalitate și continuă să producă pixeli ai unei curbe atâta timp cât aceasta nu trece prin punctul final dat.)

10,8 A POINT EDITOR

Potrivirea curbei în grafica computerizată necesită specificarea unui set de parametri, astfel încât curba calculată din ei să aibă forma dorită. Este adesea destul de dificil să selectați valorile adecvate ale acestor parametri în avans și apare necesitatea unui design interactiv. În cazul polinoamelor Bezier trebuie să avem mijloace pentru a introduce un set de puncte de ghidare și pentru a calcula și afișa polinomul. Dacă forma rezultatului nu este acceptabilă, ar trebui să putem modifica locația punctelor de ghidare, să introducem altele noi sau să ștergem unele. Un instrument de bază pentru astfel de operațiuni este un editor grafic interactiv. Vom folosi termenul de editor de puncte (spre deosebire de un editor de text sau de imagini) pentru a descrie o astfel de facilitate. Un editor de puncte poate fi folosit pentru o serie de alte lucruri în afară de potrivirea curbei. Apariția sa în acest capitol nu implică faptul că specificarea polinoamelor Bezier este aplicația sa majoră .

Există două considerații importante în proiectarea unui editor de puncte: structura de date utilizată pentru stocarea punctelor și mijloacele de introducere și adresare (indicare către) puncte.

10.8.1 O structură de date pentru un editor de puncte

Cea mai simplă structură posibilă este o matrice bidimensională a coordonatelor (x^) ale punctelor. Cu toate acestea, acest lucru face ca inserarea și ștergerea punctelor să fie foarte greoaie. Cel mai bine este să utilizați o listă legată. Elementele listei constau din patru intrări: coordonata x, coordonata y, adresa elementului anterior din listă, *(p)* și adresa elementului următor din listă, (n). Acest lucru ar putea fi implementat ca patru matrice unidimensionale cu indexul matricei folosit ca adresă.

Exemplul 10.9: Să presupunem că începem cu punctele (5,8). (11,22) și (14,18). Apoi, intrările din listă vor fi:

(5,8,0,2), (11,22,1,3). (14,18,2,0).

0 denotă un pointer către un element inexistent și astfel marchează primul și ultimul punct al listei (vezi Figura 10.8a). dacă inserăm punctul (9,10) între primul și al doilea, atunci lista va fi:

(5,8,0,4), (11,22,4,3), (14,18,2,0), (9,10,1,2).

(Figura 10.8b.) Ștergerea ultimului punct al secvenței (al treilea din listă) va avea ca rezultat

(5.8.O.4), (11,22,4,0), (14.18,3,3), (9.10,1,2).

(Figura 10.8c) unde am folosit auto-indicarea pentru punctul șters, schimbând al doilea și ultimul punct produce următoarele:

(5,8,0,2), (11,22,1,4). (14,18,3,3). (9.10,2,0).

(Figura 10.8d.)

□

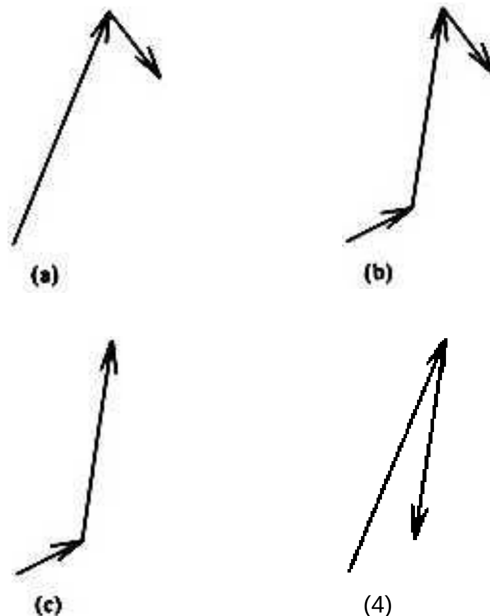


Figura 10.8 **Dispunerea punctului utilizată în Exemplul 10.9**

O modalitate și mai bună de implementare a listei legate este oferită de structurile și pointerii matrice. Din punct de vedere conceptual, manipularea datelor este aceeași ca cea prezentată mai sus, dar introduce o notație mai simplă atunci când descrieți algoritmul. Definim *punctul de structură* după cum urmează:

punct structura „ \x,y,p,n}

(Presumăm că cititorul este familiarizat cu structurile și pointerii care sunt utilizați în multe limbaje de programare.) Punctele curbei ar fi stocate într-o matrice constând din astfel

de structuri. Vom folosi valoarea 0 ca mai înainte (indicând spre nicăieri) și valoarea *L* pentru a desemna prima locație liberă disponibilă în memorie. Simbolul *q* va indica indicatorul către locația curentă.t Astfel, pentru a trasa o linie între punctul curent și cel dinaintea acestuia putem folosi următoarele comenzi (vezi Secțiunea 1.7, Tabelul 1.2):

setp((q->p)—>x,(q—>p)—>y) vec(q—>x,q—>y)

Prima comandă setează punctul curent al plotterului la punctul anterior celui indicat de *q.* A doua comandă desenează vectorul dintre cele două puncte.

Exemplul 10.10: Următoarea secvență va adăuga un punct la listă, utilizatorul indicând atât locul de inserare, cât și locația noului punct.

1. r«^(X|^|)(Utilizatorul indică un punct pe ecran)
2. Găsiți *q* astfel încât perechea *(q—>x,q—>y)* să fie cea mai apropiată de (x^i).
3. *q„ = q—>n* (Salvați adresa vechiului „punct următor")
4. *q—>n* - £(Noul „punctul următor" va fi în prima locație liberă)
5. Utilizator prompt
6. *readp{x* $_2$ ^ $_2$)(Arată spre o a doua locație de pe ecran)
7. *L—>x* — x $_2$ *L—>y = y* $_2$ (Adăugați punct la listă)
8. *L—>p = q L—>n - q„* (Legătură)
9. *q* $_H$ *-->prev* — *L* (linkul de actualizare a vechiului „punct următor")
10. Crește /.(Actualizează adresa primei locații libere) □

Pentru multe aplicații, este important să poți trata grupuri de puncte, cum ar fi arcuri. Acest lucru poate fi realizat prin introducerea unei structuri de date ierarhice. O curbă poate fi reprezentată printr-un pointer către o matrice a punctelor sale etc.

Cel mai bine este să scrieți programe care implementează editori de puncte într-o manieră independentă de dispozitiv. În special, programul principal ar trebui să fie preocupat doar de adrese (perechi de coordonate *xy*), comenzi (folosind un nume generic) și, desigur, nume de fișiere. Toate procedurile în funcție de dispozitiv trebuie să fie separate și identificate clar ca atare. (Consultați următoarea subsecțiune pentru mai multe despre acest subiect.) Următoarea listă de

t Notația $q \longrightarrow u$ denotă obiectul tabloului de structură u la care indică indicatorul q.

comenzile este tipică Fiecare comandă necesită un set de puncte (numărul acestora este afișat în paranteze în listă) sau un nume de fișier (dacă litera / apare în paranteze). Literele aldine denotă abrevieri de comandă.

^oi(O)	Trasează datele disponibile din listă.
bezierfO}	Trasează un polinom Bezier folosind elementele listei ca puncte de ghidare.
tppend(O)	Adăugați puncte la sfârșitul listei sau creați o nouă listă dacă lista curentă este goală.
inserif 1}	Introduceți puncte înaintea celui adresat.

^elcie{ 1)	Ștergeți un punct din listă.
repeta] I)	Introduceți un nou punct în listă cu coordonatele identice cu cele adresate. Această comandă poate fi folosită pentru a crește multiplicitatea punctelor de ghidare.
muta (2)	Mutați punctul adresat în locația celei de-a doua adrese.
**riie(f)*	Scrieți conținutul listei pe fișierul numit.
citește (J)	Citiți conținutul fișierului în listă.

Comenzile *insert* și *append* presupun că utilizatorul furnizează o secvență de puncte care sunt adăugate la listă. De asemenea, este necesară o comandă specială *de evadare* pentru a indica când procesul este terminat.

10.8.2 Intrare și ieșire pentru un editor de puncte

Funcția *readp(x^)* descrisă în Secțiunea 1.7 (Tabelul 1.2) poate furniza editorului coordonatele punctului. Cu toate acestea, mai sunt multe de făcut înainte de a putea selecta o comandă sau o adresă. Unele dispozitive de indicare conțin taste (butoane) care permit utilizatorului să transmită unul sau mai multe caractere în plus față de coordonatele punctului, dacă numărul de caractere este suficient de mare, atunci ele pot fi folosite pentru a codifica comenzi . Dacă nu este cazul, atunci se poate folosi un *meniu:* o zonă a ecranului este pusă deoparte și subdivizată în regiuni care sunt etichetate cu numele unei comenzi. Când valorile *xy* primite de program se încadrează în una dintre aceste zone, se selectează comanda corespunzătoare.

O a doua problemă este să selectați din listă un punct care este cel mai apropiat de punctul indicat. (Acesta este cazul când utilizatorul dorește să selecteze unul dintre punctele existente). O soluție naivă este căutarea minimului distanței euclidiene dintre punctul dat și cele din listă. O modalitate mai eficientă este de a menține o structură de date auxiliare în care punctele sunt sortate în funcție de locațiile lor.

Algoritmul 10.3 prezintă un editor de puncte minime. Se presupune că, pe lângă rutinele grafice din Tabelul 1.2. sunt disponibile și următoarele rutine:

comanda! x,y,c) care returnează un șir *c* care conține un nume de subrutină sau un șir nul dacă (xy) nu indică nicio comandă.

locul (xyq) care localizează punctul cel mai apropiat al listei de (x^). *q* este valoarea indicatorului pentru locația găsită.

Editor de puncte Algoritm 10.3

1. **Repetați** (la infinit!) pașii de la 2 la 5.

ÎNCEPE.

2. *readp(xy).*

3. de apel *(x jc).*

d- Dacă *c* este nul sau dacă subrutina *c* necesită o adresă, atunci apelați *locul (xy.q).*

5. Sună *c.*

Sfârşit.

6. **Sfârșitul algoritmului.**

Algoritmul 10.3 presupune că este disponibilă o comandă *de ieșire* pentru a ieși din buclă. Valorile lui x și *y* returnate la pasul 2 ar trebui să fie introduse într-o stivă pentru o posibilă utilizare ulterioară.

6.9 NOTE BIBLIOGRAFICE

Majoritatea manualelor de analiză numerică descriu tehnici de interpolare polinomială (de ex. 110.IKI). Există, de asemenea, cărți dedicate în întregime subiectului (de exemplu (10.DAJ). Un tratament matematic extins este dat în |10 RI). Polinoamele Bezier poartă numele unui matematician francez care le-a dezvoltat pentru producătorul de automobile Renault (IO.BE). Acestea sunt discutate în multe dintre textele care se ocupă de grafică, deși se acordă puțină atenție (de obicei) prezentării unor algoritmi rezonabili pentru calcularea lor. Teorema]0.1 a fost Uken din I10.LR], în timp ce proprietatea de segmentare mai generală poate fi găsită în (10 BE). Formulele pentru arcele circulare se bazează pe geometrie analitică simplă și au fost incluse aici datorită interesului larg în utilizarea lor.

Secțiunea de afișare ar fi putut fi mult mai lungă, deoarece reprezentarea efectivă a curbelor pe o grilă discretă este o problemă cu adevărat provocatoare . Cititorul este referit la I10.BR] și (10.KU) pentru informații suplimentare despre afișarea arcurilor și la [2.GSI. [2.KU). I10.CR] și [IOWA] pentru utilizarea scării de gri pentru a îmbunătăți aspectul vectorului

afișează în grafică raster.

Problemele de stabilitate numerică sunt tratate în majoritatea textelor de analiză numerică. În special, problemele de divergenţă cauzate de aproximările discrete pot fi studiate cu metodele prezentate în textele de soluţionare numerică a ecuaţiilor diferenţiale şi cărţile despre ecuaţiile diferenţiale (ex.[IO.LL)).

Editorul de puncte descris în acest capitol este destul de modest și majoritatea instalărilor grafice oferă mult mai mult. O revizuire recentă a chestiunii de editare poate fi găsită în (10.CG).

6.10 LITERATURA RELEVANTA

(I0.BE1 Bezier. P. *Numerical Control.* **New York: J. Wiley. 1972. (Trans), din franceză de AR Forrest și AF Pankhurst.)**

[10.BR] Bresenham. J. „Un algoritm liniar pentru afișarea incrementală a arcurilor circulare.” *CACM,* **20 (1977), p. 100-106.**

IIO.CG] Coueignoux. P. și Guedj, R. „Generație computerizată de modele plane colorate pe rasterele TV-Like”, *IEEE Proceedings,* **68 (1980). p. 909-922.**

(10.CR) Crow, FC „Utilizarea tonurilor de gri pentru afișarea raster îmbunătățită a vectorilor și a caracterelor” *SIGGRAPH 78,* **pp. 1-5.**

(10.DA) Davis, PJ *Interpolation and Approximation,* **New York: Random House. Blaisdell, 1963.**

[10.IK] Isaacson. E. și Keller, HB *Analysis of Numerical Methods,* **New**

York: J. Wiley. 1966.

[10.KU) Kulpa, Z. „On the Properties of Discrete Circles, Rings, and Disks" *CGIP,* **10 (1979), pp. 348-365.**

[I0.LL] Levy. H. și Lessman, F. *Ecuații cu diferențe finite,* **New York: Macmillan, 1961.**

[10.LR] Lane. JM și Riesenfeld, RF „A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces" , *IEEE Trans on Pattern Analysis and Machine intelligence.* **PAMI-2 (1980). pp. 35-46.**

(I0.RI] Rice, JR, *The Approximation of Functions,* **vol. 1 și 2. Reading. Mass.: Addison-Wesley. 1965 și 1969.**

[IO.WA] Warnock. J, F. „Afișarea caracterelor utilizând matrice de mostre de nivel de gri". *SIGGRAPH'80.* **pp. 302-307.**

6.11 PROBLEME

10.1. Dorim să trecem o curbă netedă printr-un set de puncte [x/^}

în avion. Se propune următoarea soluție. Considerați punctele ca vârfurile unui poligon și pentru fiecare *i* găsiți trixa bisec a unghiului la acel vârf. Luați normala bisectricei care trece prin vârf ca tangentă dorită a curbei în acel punct și utilizați ecuația (10.5) pentru a defini un arc cubic care unește fiecare pereche de puncte.

(a) Fie *Nx,* să notăm diferența xr^-i împărțită la lungimea vectorului de la (x/.ı^-ı) la (x (j» ( ), și să fie *Ny t* diferența similară pentru *y.* Apoi arătați că ecuația normalei la bisectrică la (x^)) este:

(Mq+I-AMU -*) + Wui -AtoMy -*) -o

(b) Utilizați expresia de mai sus din ecuația (10.5) pentru a obține o formă explicită pentru arcul cubic. Faceți acest lucru atât pentru forma y "/(x), cât și pentru reprezentarea parametrică x — x(r) și j " y (r) unde 0<r SI.

(c) Examinați avantajele și dezavantajele fiecărei forme și deduceți o condiție asupra formei poligonului astfel încât curba netedă dată de reprezentarea parametrică să nu se încrucișeze.

10.2. Demonstrați ecuația (10.10a).

10.3. Scrieți un program care implementează algoritmul 10.1.

10.4. Scrieți un program care implementează algoritmul 10.2 și comparați timpul de rulare al acestuia cu cel al programului scris pentru exercițiul anterior. Trasează acești timpi ca funcții ale lui m. Cum se compară timpii observați cu predicțiile teoretice? Cu ce atribui discrepanța?

10.5. Demonstrați a doua parte a teoremei 10.1 (pentru 0.5<r < I).

10.6. Scrieți un editor simplu care încorporează o comandă pentru trasarea unui polinom Bezier ca una dintre opțiunile sale. (Puteți folosi câteva dintre implementările de mai sus în acest scop.) Ridicați un set de puncte care formează conturul unui obiect fizic și încercați să generați un polinom Bezier care va trece în apropierea acestor puncte. Acest lucru necesită un design interactiv,

deoarece selectarea punctelor de ghidare nu este deloc evidentă. Astfel, editorul este o parte esențială a proiectului. *(Notă:* Încercați să aranjați programul astfel încât să puteți adăuga alte caracteristici la acesta în viitor.)

10.7. Comparați numărul de biți necesari pentru stocarea coeficienților

a polinomului Bezier aproximând conturul unui obiect cu numărul de biți necesari pentru stocarea codului de lanț diferențial al aceluiași contur. Repetați pentru câteva exemple. Puteți trage concluzii generale?

10.8. Având în vedere descrierea parametrică a unei curbe, *X(t), Y(t)* și un număr *N,* stabiliți o procedură de alegere a densității punctelor de eșantion pe intervalul (0,1], astfel încât o afișare a curbei prin vectori care unesc *N* puncte de eșantionare să fie cât mai lină posibil.

10.9. Dorim să afișăm funcția *ax"* pe un dispozitiv grafic raster. Deoarece *a* nu este un întreg și dorim să facem calculul numai cu aritmetică întregi, decidem să trasăm *bx"* unde *b* desemnează numărul întreg cel mai apropiat de *a.* Presupunem că atât *a* cât și *b* depășesc 1, la fel și *n.* Pentru a avea o diagramă continuă trebuie să activăm mai mult de un pixel pentru fiecare valoare a lui x. Puteți găsi o regulă pentru a face acest lucru, astfel încât valoarea rotunjită a lui *ax"* să fie întotdeauna pe un astfel de pixel? Dacă credeți că acest lucru este imposibil, furnizați o dovadă formală pentru afirmația dvs.

10.10. Scrieți un program pentru a desena o linie dreaptă între o pereche de puncte fără a utiliza o comandă grafică. Repetați pentru un cerc.

10.11. Arătaţi că cercul definit de punctele necolliniare (Xjj'i), (*2^2). iar (XJ^J) are un centru *(xc,yc)* dat de

x ^i(^rh) - *C* $_2$ *(y* $_t$ *-y* $_2$ *) (X'-xd*^*-yd* - (xj-XjKy,-^) (10.48a)

Czfxi^xz) — C|(x $_2$ —x $_3$)

(X|-x $_2$)(y $_2$ -y $_3$) - (x $_2$ -x $_J$)O'|-y $_2$) (10.48b)

unde

Cl " (*|“X $_2$)(X)+X $_2$) + (yry $_2$)(yi+y $_2$) și

C2- (XJ-XJKXJ+XJ) + (y2 - yj)(y2 + yj) .

Arătați că numitorul ecuațiilor (10.48) va fi zero, cu excepția cazului în care cele trei puncte nu sunt coliniare.

Capitolul 11

MONTARE CURBA CU SPLINES

11.1 INTRODUCERE

În multe aplicații în care se utilizează potrivirea curbei, s-ar dori să modifice părți ale curbei fără a afecta alte părți. Vom spune că o schemă are o proprietate locală dacă modificările locale nu se propagă. În mod clar, polinoamele discutate în Secțiunea 10.2 nu au această caracteristică, iar polinoamele Bezier o prezintă doar aproximativ. O modificare a locației sau a multiplicității unuia dintre punctele de ghidare necesită recalcularea întregii curbe, chiar dacă modificările vor avea un efect redus departe de punctul de ghidare modificat. Funcțiile polinomiale pe bucăți oferă o modalitate directă de a obține controlul local. Vom discuta astfel de funcții mai întâi în forma y -y(x) și mai târziu în reprezentări parametrice. Următoarea este o expresie generală pentru o funcție polinomială pe bucăți:

p(x)-p,(x) x (<x<x, +1 f-0, I, •••.4-1 (11.la)

A 0, (*J“AW(Xi) 7^ /-I. •••.4-1 (II.lb)

Punctele x b • • , x*_| care împart un interval (ob] în 4 subintervale se numesc de obicei puncte de întrerupere. Punctele curbei la aceste valori ale lui x se numesc de obicei noduri. Pentru comoditate se folosește notația x 0 " a și x* — 6. Funcțiile ^(x) sunt polinoame.

de gradul *m* sau mai mic. Constrângerile de continuitate la punctele de întrerupere sunt exprimate prin al doilea set de ecuații unde p/ 0> U) reprezintă A(x). și *p t ui (j>W* pentru lhc /* derivată a lui *p(x).* Uneori vom folosi valoarea r - 0 pentru a indica absența oricăror constrângeri. Când r - I avem o funcție continuă, dar fără constrângeri asupra derivatelor sale. Dacă r - m+l, atunci avem un singur polinom peste *la .b]* astfel încât *r = m* este numărul maxim de constrângeri care dă o funcție polinomială netrivială pe bucăți. Cazul cu m — 3 și *r* — 3 are o semnificație istorică, precum și practică deosebită, iar funcțiile polinomiale pe bucăți corespunzătoare sunt cele pentru care a fost folosit pentru prima dată termenul splines.

Secțiunea 11.2 este o introducere la spline și Secțiunea 11.3 la B-splines. Secțiunile 11.4 și 11.5 tratează probleme de calcul. Secțiunile 11.6 și 11.7 arată cum splinele pot fi utilizate în mod avantajos în grafică.

11.2 DEFINIȚII FUNDAMENTALE

Originea termenului de spline se încadrează în zilele dinaintea graficii pe computer, când desenatori obișnuiau să localizeze greutăți la punctele de date și apoi să plaseze o riglă flexibilă din lemn, numită spline, împotriva greutăților, pentru a crea o curbă netedă care trecea prin puncte. Greutățile aveau o proeminență ieșită în afară care se potrivea într-o fantă a splinei și o ținea pe loc, permițându-i în același timp să se deplaseze în jurul punctului fix. Este posibil să se utilizeze teoria elasticității mecanice și să se demonstreze că curba rezultată este (aproximativ) un polinom cubic pe bucăți care este continuu și are derivate prima și a doua continue. Aceste condiții asigură, de asemenea, că curba are o curbură continuă și discontinuitățile apar numai în derivata a treia. Deoarece este foarte dificil pentru ochiul uman să distingă pe acesta din urmă, curba rezultată pare complet netedă. Dacă direcționăm o mișcare mecanică de-a lungul unei spline, o derivată a doua continuă implică o accelerație continuă și, prin urmare, nicio modificare bruscă a forței. Aceste două proprietăți fac astfel de curbe foarte de dorit pentru multe aplicații practice. Cu toate acestea, există cazuri în care cineva poate fi mulțumit cu mai puține constrângeri de continuitate sau polinoame de ordin inferior și termenul spline a fost folosit pentru a acoperi și aceste cazuri. De fapt, astăzi există puține acorduri cu privire la utilizarea termenului. În acest text vom folosi următoarea terminologie:

Definiția 11.1: O *spline simplă* este o funcție polinomială pe bucăți, dată de ecuațiile (11.1) cu *r • m.* Termenii spline liniar, spline quad ratic și spline cubic corespund valorilor lui m - L 2 și 3. □

Definiția 11.2: O *spline* este o funcție polinomială pe bucăți, dată de ecuațiile (11.1) cu $r < m$. *O*

Putem observa că funcțiile utilizate în Exemplul 10.2 (Figura 10.lb) și Exemplul 10.3 (Figura 10.2) sunt spline pătratice, în timp ce curba problemei 10.1 este o spline cubică (vezi și problema 11.1). În mod clar, se pot avea spline atât de interpolare, cât și de aproximare. În ambele cazuri este important să se evalueze numărul de grade de libertate pe care le are curba. Ecuațiile (11.1) arată că avem A(m + 1) coeficienți polinomiali minus (^~l)r constrângeri pentru un total de *k(m—r)+k+r* grade de libertate pentru o spline. Pentru o spline simplă unde *m—r avem* doar *k + m* grade de libertate.

Majoritatea lucrătorilor din această zonă sunt de acord că cea mai importantă problemă în utilizarea cu succes a splinelor este alegerea numărului și a locației punctelor de întrerupere. Cititorii se pot mulțumi de importanța acestei întrebări analizând exemplele din Secțiunea 10.2. Se pare că aceasta este o problemă mult mai dificilă decât găsirea unei spline de interpolare sau de aproximare după ce punctele de întrerupere au fost definite și vom amâna discutarea ei în detaliu până la capitolul următor, limitându-ne aici la câteva observații preliminare. Observăm că dacă permitem

ca punctele de întrerupere să se unească, adică au x_t — $x_{/+}$ |. atunci numărul de constrângeri este redus automat cu una. Dacă avem puncte de întrerupere cu multiplicitatea r+l, atunci toate constrângerile sunt eliminate. Introducerea mai multor puncte de întrerupere pentru a produce spline nesimple are atât avantaje teoretice cât și practice și vom prezenta câteva dintre ele mai târziu. Mulți autori folosesc, de asemenea, termenii spline cu noduri simple și spline cu noduri multiple pentru a desemna spline simple și, respectiv, spline. Pe de altă parte, există probleme pentru care este mai convenabil să se permită ca r să fie diferit de m și, eventual, diferit la fiecare punct de întrerupere.

O altă problemă majoră în aplicarea spline-urilor este forma matematică folosită pentru curbă. Ecuația (11.1) conține prea mulți parametri, iar utilizarea constrângerilor pentru a le elimina are ca rezultat o analiză destul de complicată. (Se poate demonstra că constrângerile sunt într-adevăr independente liniar, astfel încât toate sunt eficiente.) O formă alternativă este dată de următoarea ecuație:

?(x) -Pi(*) + S 4/ (*~*i)r OI-2) ii

unde funcția l^ reprezintă r" când />0 și 0 când f <0. Observăm că r* și primele sale m— 1 derivate sunt zero pentru / — 0, astfel încât *p(x} și primele sale m—* [1] derivate sunt continue la toate punctele de întrerupere. Termenul q_t este proporțional cu cantitatea de discontinuitate a *m'k*

derivată introdusă la *x*< și deci funcția definită de ecuația (11.2) are aceleași proprietăți ca și cea dată de ecuația (11.1). pentru *r „ m* .

Pj(x)-p $_t$(x) + £q, (xx^T. (11.3) /-i

Ecuația (11.2) are doar (w+1) + (*-l)-m+i parametri liberi , numărul minim. Forma sa seamănă cu cea a unui polinom, iar acest lucru îl face destul de convenabil pentru multe studii teoretice. Din pacate, are anumite defecte grave. În primul rând, Ecuația (11.3) sugerează că dacă dorim să modificăm funcția la un subinterval, adică, să schimbăm valoarea lui 9, la capătul său din stânga, atunci reprezentarea trebuie schimbată la toate subintervalele ulterioare. Pe de altă parte, ecuația (11.1) arată că nu este necesar să fie așa. De exemplu, dacă m =1, atunci forma lui p(x) trebuie schimbată în doar două subintervale (vezi Figura 11.1a). În mod similar, dacă m = 2, numărul de subintervale nu trebuie să depășească patru. Într-adevăr, în exemplul din figura 11.1b, s-ar putea decide să mențină locația și tangentele la nodurile /I și *E* fixe, astfel încât restul splinei să fie izolat de schimbare. Dacă locația (dar nu și tangentele) nodurilor *B* și *D* se păstrează, de asemenea, aceeași, atunci arcele parabolice care formează spline dintre *A* și *B* și *D* și *E* sunt complet specificate. Apoi, pentru spline cu noduri la *B, C'* și *D* avem ca date cele trei noduri, cele două tangente la capăt și constrângerea de continuitate la *C* pentru un total de șase constrângeri și există, de asemenea, șase grade de libertate. Deoarece reprezentările de pe fiecare subinterval comunică între ele prin constrângerile de la punctele de întrerupere, se așteaptă ca, în general, numărul de subintervale afectate să fie proporțional cu numărul de constrângeri. Următorul rezultat, care se ocupă de introducerea unui nou punct de întrerupere în subinterval ($x_{(l}$ $x_{(+1}$))

prezintă un exemplu al acestei observații într-o manieră riguroasă.

Propunerea 11.1: Dacă se introduce un nou punct de întrerupere în subinterval (x b x)+ |), atunci spline-ul trebuie modificat numai pe subinterval *(x it* x f+ „).

Dovada: Fie *u* numărul de subintervale (originale) afectate. Potrivirea valorii spline și a primelor sale derivate m—1 la *x t* și *x t* + *u* impune constrângeri de 2m. Numărul total de subintervale disponibile este u+1 pentru un total de m+u+1 grade de libertate. Prin urmare, u " m—1. □

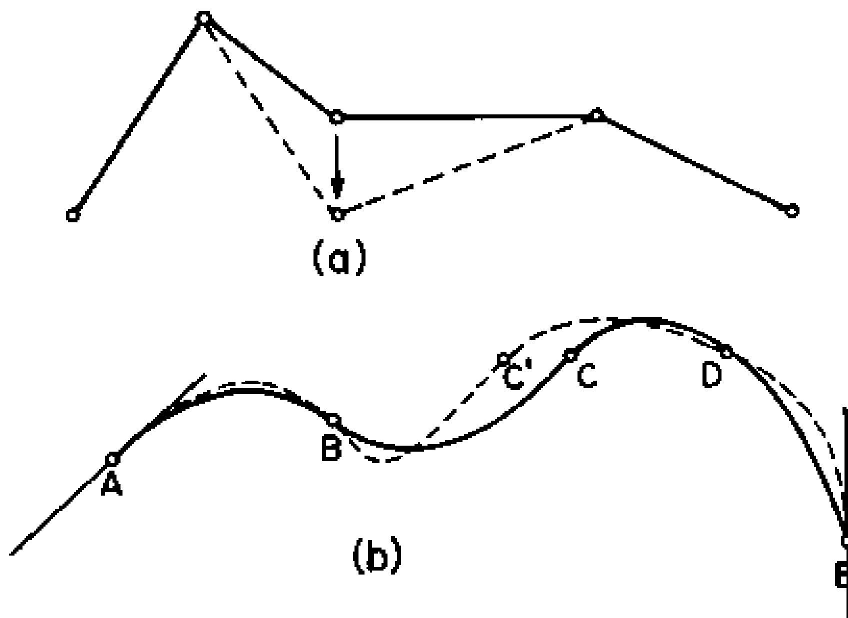


Figura 11.1 **Ilustrarea naturii locale a splinelor: (a) când m — I schimbarea locației unui nod necesită reevaluarea spline-ului numai în cele două subintervale adiacente, (b) când m - 2 vor fi patru subintervale afectate.**

Pentru *m* „ 1 nu trebuie să modificăm niciun alt subinterval, în timp ce pentru m „3 este posibil ca cele două subintervale ulterioare să fie modificate. Dacă numărul subintervalelor nu este afectat, dar introducem o constrângere suplimentară într-unul dintre ele, un argument similar poate fi folosit pentru a arăta că M =m+l. Cititorul poate fi nedumerit de ce trebuie să modificăm numai subintervalele după schimbare și nu pe cele dinaintea acesteia. Răspunsul la puzzle devine evident dacă distingem între modificările curbei în sine și modificările reprezentării curbei. Astfel, Propunerea 11.1 se referă la o modificare a reprezentării curbelor.

Un al doilea dezavantaj al ecuației (11.2) este că tinde să facă anumite probleme de aproximare instabile numeric. Din aceste motive, o a treia formă de reprezentare spline este utilizată în multe aplicații practice. Spline-urile sunt exprimate ca sume ale altor spline și, în special, o formă specială numită B-spline.

11.3 B-SPLINES

B-splines-urile sunt spline care sunt zero la toate subintervalele, cu excepția m+1 dintre ele. Figura 11.2 prezintă exemple de B-spline liniare, pătratice și cubice. Astfel de spline pot fi definite recursiv după cum urmează:

Definiția 11.3: B-spline constantă peste subintervalul /'* este definită ca

1 x^x^+J

^oC*) (IM)

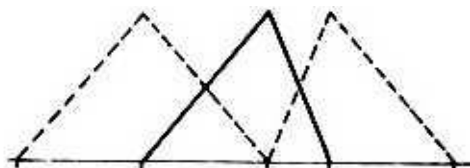
0 otherwise

m '* pe intervalul (x /f x, +m+ |] este definit ca

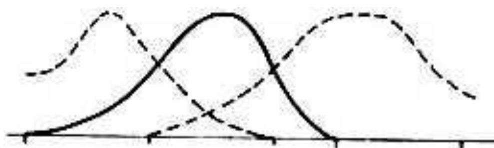
^<'> - «^(x) + x x'_+"V "<*>--.(*) (11.5)

*r+m x t x /+«+i *r+i

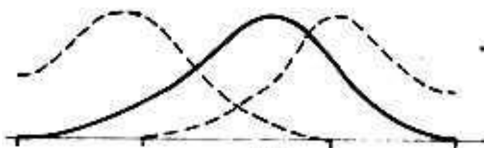




(o)



(b)



(c)

Filire 11.2 Exemple de B-spline: (a) liniară, (b) pătratică, (c) cubică

Putem folosi ecuațiile (11.4) și (11.5) pentru a găsi forme explicite pentru B-spline de grad inferior.

Liniar

^,i(x) **X ~X,** */+l “ */ *i+2 ~ * Xf+2 ~ **x/+l<x <XH2** (H-6)

Quadratic

A«*) “ **(X** **(^H-XfXXHi - Xi)** **for x(£x<x(+|** (11.7a)

A«*) **(X -XjCXnj-x)** **(*/+2-Xi)(XJ+2-Xr+1)**

t (xm-x)(x-xH) **fori|(|<x <xl+2** (H.7b)

(*(+3 ~ X/+|)(Xf+2 — X/+|)

AU*) **for x/+?<x<x/+3** (11.7c)

(*/+3 *f+l)(*f+3 X/+2)

Când punctele de întrerupere sunt egal distanțate la intervale de lungime *L.* expresiile de mai sus sunt simplificate semnificativ. În acest caz, este rezonabil să presupunem că x, ■ *iL* şi convenabil să introducem variabila normalizată

u **x ~ x, x** *L ~ L* (H.8)

Apoi ecuațiile pentru B-splines devin

Liniar uniform

u 0<u<l

IVO-MD- 2 _ u l£a£2 (H.9)

Quadratic uniform

U h2 Ui+u)L)- 1“ J pentru 0<«<! (11.10a)

l-<'-p

UKIW+UK)- pentru 1<«<2 (11.10b)

U(' + “) £)" pentru 2<u <3 (11.10c)

y(3-<0 2

Ecuația (11.5) poate fi utilizată direct pentru cazul distribuției uniforme a punctelor de întrerupere și în acest fel putem găsi expresiile pentru B-splinc-ul cubic din cele ale B-splinc-ului pătratic.

U13 ((I+U)L)-~YU3 FORO < M<l (11.11a)

£/,.,((/+«)£)--|—y(«-2) ' -(«-2) 2 forl£u<2 (11.11b)

l/ (.,((i+u)Z.)-y+y(u-2)'-(«-2) 2 pentru2£u£3 (11.11c)

(/,,((i+u)£)-7(4-u) ! pentru3£u4£ (11.capac)

Folosind B-spline-urile ca bază, putem exprima orice spline ca

P(x)= 2 «A^^) (H.12)
r—m

Această ecuație conține exact parametrii Hw: *a^ m ,* o- m+ i, ■ • • a 4 _j. Pe fiecare subinterval, *p{x)* este dat ca suma a cel mult m+J B-spline, astfel încât prezintă într-adevăr comportament local. Modificarea oricăruia dintre coeficienții din ecuația (11.12) va schimba forma curbei în doar m + 1 intervale. Următorul exemplu ilustrează unele dintre proprietățile importante atât ale spline-urilor B, cât și ale splinelor în general.

Exemplul 11.1: Dorim să găsim o spline pătratică de interpolare pentru următoarele puncte de date: (0,0), (1,1), (2,2), (3,?), (4,2). (5,1), (6,0). Vom investiga efectul valorii lui P asupra aproximării utilizând B-spline uniforme cu L — l. Înlocuind expresiile din ecuația (11.10) în ecuația (11.12) și înlocuind w cu definiția sa din ecuația (11.8) (cu *L* — 1) găsim

P(x) -yaAx-f)^

^-i^ „ (x-(* + y)] 2 } + y^-zO+lx) 7 f <x<f+l (ll.13a) și pentru derivată pe același interval

p'(x) ~ a,(xi) - ^-^-(Z+y)] - ^-jd+lx) (11.13b)

Ambele ecuații sunt valabile pentru f — 0, 1, • • • n—l și dau următoarele valori la punctele de întrerupere.

P<0 “ yk<-i + U H2] p'(f)= ^-I-^-J

(11.14a)

Pentru interpolarea actuală a ecuațiilor:

(11.14b)

oi + a-2 " 0 «o + O-|

Oi + oo ”4 o 2 + a (°3 + o 2 “ 4 o 4

+ o 3 Os + a 4 “0

problem we have the following set

“2

- IP

- 2

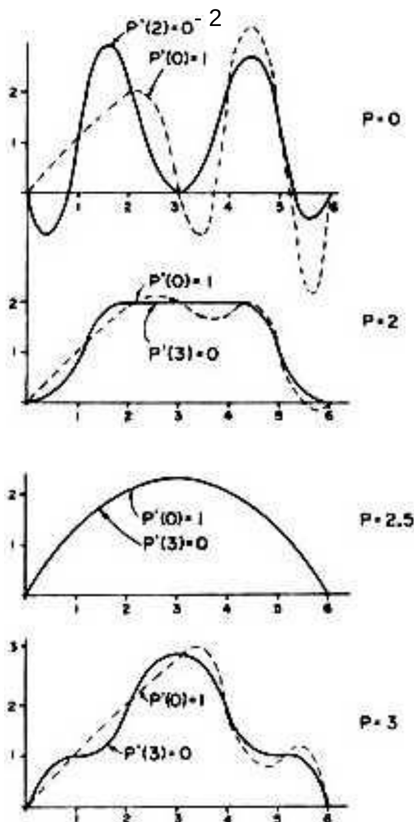


Figura 11.3 **Ilustrație pentru Exemplul II.I: Efectele locației unui punct (valoarea lui** *P).* **și specificarea unei tangente pe forma unei spline pătratice**

Există mai multe necunoscute decât ecuații și putem alege să specificăm tangenta la un moment dat. Vom studia două cazuri:

Caz (a}: Alegeți *p'(3)* - 0, adică o tangentă orizontală la mijlocul setului simetric de puncte de date. Găsim

o_ 2 = o $ " 2-P *a_ x - a 4 = ~2+P*

***0 = *3" 4-P** ***1 „ fl 2 = p**

Toți coeficienții depind de *P,* astfel încât modificarea valorii unui punct de date afectează întreaga aproximare spline. Figura 11.3 ilustrează patru cazuri: P " 0, 2, 2.5 și 3.

Cazul (b): Alegeți p'(0) = L Găsim acum un interpolant asimetric cu

a_ 2 " —0,5 a_i — 0,5 a 0 —1,5 a, —2,5

a 2 = 2P-2.5 a 3 = 6.5-2P o 4 = -4.5+2P a 5 = 4.5~2P

Il este ușor de verificat că *p(x)* este egal cu x în intervalul (0,2) și, prin urmare, valorile spline-ului pe primele două intervale nu depind de *P.* Totuși, această îmbunătățire a cauzat pierderea simetriei. Graficele lui p(x) pentru patru valori ale lui *P* sunt de asemenea prezentate în Figura 11.3. □

Trebuie subliniat faptul că rezultatele finale (adică, valorile lui p(x)) sunt independente de alegerea reprezentării pentru spline și ar fi fost la fel dacă am fi avut-o. de exemplu, a folosit ecuația (11.2). Exemplul 11.1 și Figura 11.3 ilustrează câteva proprietăți ale spline. Deși interpolările spline sunt locale, alegerea pentru satisfacerea gradelor suplimentare de libertate poate avea un efect major asupra rezultatului final. Acest lucru nu contrazice de fapt Propunerea 11.1, care afirmă pur și simplu că putem modifica curba la nivel local prin introducerea unor puncte de întrerupere suplimentare. Cititorul poate încerca, ca exercițiu, să introducă două puncte de întrerupere în loc de unul în intervalul (2,4) în exemplul de mai sus (vezi problema 11.4). O a doua concluzie este că spline-urile pot prezenta oscilații între punctele de date la fel de severe ca cele cu interpolări polinomiale. Motivul pentru care aproximările pe bucăți au dat rezultate bune în exemplele din Secțiunea 10.2 este că a existat o oarecare libertate în selectarea punctelor de întrerupere. Am fi putut obține rezultate mai bune în exemplul de mai sus dacă am fi introdus puncte de pauză suplimentare. Un calcul simplu poate verifica că locațiile 2.3 și 3.7 cu *p'(x)* = 1 și respectiv —1 când P = 3 dau aproximații liniare peste (0,2) și (4,6). Același rezultat se obține cu locațiile 2.5 și 3.5 când P = 0. (În acest caz, alegerea potrivită pentru *p'(x)* este *p'{2)* - 1 și p\4) --|.)

Încheiem această secțiune cu o proprietate interesantă a B-splines.

Teorema 11.1: Pentru toate valorile lui x și *m*

$$2^Wi. \qquad (11.15)$$

Notă: Limitele de însumare nu sunt date în mod explicit deoarece suma are doar m+1 termeni pentru fiecare valoare a lui x.

Dovada: Vom folosi ecuația recursivă (11.5). Acest lucru poate fi scris pentru r — eu mai degrabă decât *i* să cedeze

^-t^U) - r ^-■ J .- l (»> + -^-^.-.(x) . (11.16) X i+*-1 x fl x (+m —X/

Adăugând această ecuație la ecuația (11.5), aflăm că

^-^(x) +/V,^(x) =

xl+m-rxi-l JV^_I(x)+M^,(x) +

^^^-.(x). (11.17)

Procesul care a produs ecuația (11.17) din ecuațiile (11.5) și (11.16) poate fi repetat de la, de exemplu, f — j la f — A pentru a obține

/-* X—X/

PW*) - ; J ^-,(^>

lj x j+m x J+\

+ SA^-H*) + -^^^/W,,-^) . (11.18)

f-j+l x *+m+l x *+l

Luând *j* suficient de mic și A: suficient de mare, ne putem asigura că A^-iCx) și A^+i^-ifx) sunt zero. Atunci ecuația (11.18) poate fi scrisă ca

2^(»)-2«J.-l(ˀ). (11.19)

i i

Cu alte cuvinte, suma B-spline-urilor la un punct x este independentă de valoarea lui m. Se poate observa cu ușurință din ecuația (11.4) că valoarea sa pentru m „ 0 este I și aceasta încheie demonstrația □

11.4 CALCUL CU B-SPLINES

Ecuația (11.5) sugerează o procedură de calcul simplă pentru evaluarea unei B-spline la un punct x. Observăm că pe orice interval dat [x (> x /+ |] există doar m+1 B-spline de gradul *m* care sunt diferite de zero. Pe acel interval, *N l>m (x)* depinde numai de /^-^x) deoarece

/Vf+j^-iCO este zero acolo, în timp ce *ty-t^x) (Q<l^m)* depinde atât de ^^^.((x) cât și DE ^-/ J HI(X) Interdependența B-splines este prezentată în Figura 11.4. Pentru a găsi spline de gradul *m,* trebuie să găsim m-1 nivelurile anterioare în care trebuie să găsim liniile la fiecare nivel B din diagrama prezentată în diagrama *t* - *p care trebuie să găsim* *j(x)* la *N^ij^x),* unde *j* este gradul la acel nivel și *I* variază de la 0 la *j.* Acest lucru duce la ritmul Algo

Jl.la.

Numărul de înmulțiri și împărțiri executate de algoritm este proporțional cu m ^{2} în fiecare punct *x.* Dacă locația punctelor de întrerupere este uniformă și fixă pentru o clasă de probleme, atunci se pot calcula valorile direct din formulele explicite ale ecuațiilor (11.9) la (11.11) sau altele similare. Această abordare necesită și un efort proporțional cu m^2 . dar operatiile folosite sunt mai simple.

lu

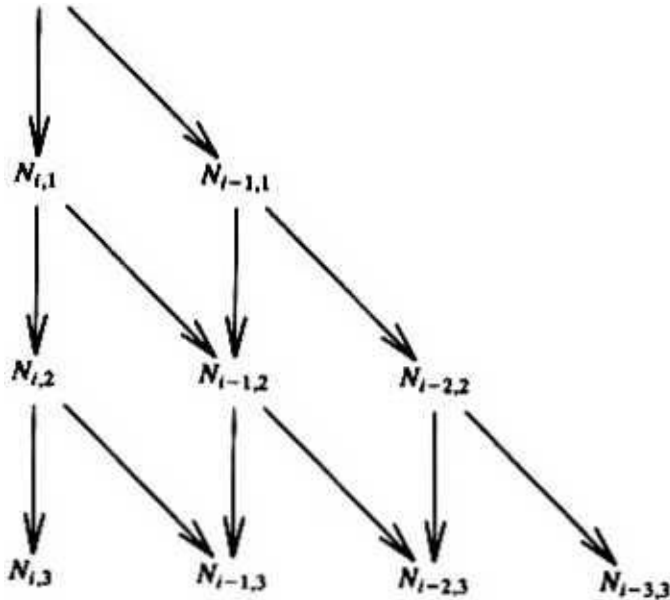


Figura 11.4 **Interdependența valorilor B-spline-urilor la un punct x. Fiecare termen este suma ponderată a unuia sau a doi termeni din linia de deasupra acestuia. Liniile cu săgeți indică fluxul de calcul. Liniile verticale indică înmulțirea cu primul factor din ecuația (11.5) și diagonalele cu al doilea factor din ecuația respectivă .**

Algoritmul 11.1a Procedura *BSPUNE(i ,x,m):* Evaluarea tuturor fl- spline la un punct x aparținând intervalului [x/^+J.

Notație: m **este gradul spline-ului, x este punctul în care sunt evaluate spline-urile. Tabloul** *NUJ}* **conține valorile** *N,j(x). a* **și** *b* **sunt variabile auxiliare.**

1. Inițializați /V(/.O) la 1.
2. **Pentru J - 1 la m faceți:**
   ÎNCEPE.
3. **Pentru / -** 0 la ; **do.**
   ÎNCEPE.
   {Calculează *N(i—IJ).]*
4. *a - ((xx ı_ ı)/(x ı_t+j~x ı_ ı))XN(i-lj-\).*
5. *b - ((xt-^+i-xMx^j+i-Xt-HMW(i-Mj-\).*
6. *N(il,j)-a +b.*

Sfârşit.
Sfârşit.
7. **Sfârșitul algoritmului.**

11.5 INTERPOLAREA B-SPLINES

Fie (/|j*i). (hj'z). ' ' ' (<•*>'•) să fie punctele de date pentru care trebuie găsită o spline de interpolare. Există o serie de modalități de a rezolva această problemă. Una este să identificați fiecare punct cu un nod spline. Deoarece o spline are *k+m* grade de libertate, dacă *m* este mic (cum este de obicei cazul), atunci se poate alege *k* — **fl—1** și se pot identifica punctele de întrerupere cu h^2» ' * ' ^w Pentru o spline liniară, unde m — 1, avem exact n grade de libertate, iar curba este complet specificată ca mulțimea de drepte care unesc punctele (f^) la (Z 2, n^) la (Z 2, n^) la (Z 2, n^^) — Eu. Pentru m — 3 numărul total de grade de libertate este n+2, ceea ce lasă două grade în plus după ce curba este constrânsă să treacă prin punctele de date. În practică, acestea sunt folosite pentru a specifica tangentele la punctele de capăt.

O altă abordare este să intercalați punctele de întrerupere și punctele de date, eventual setând *k+m — n.* Vom studia acest caz în detaliu folosind B-spline pentru a găsi coeficienții splinei de interpolare. Să presupunem că avem 4—1 puncte de întrerupere așa cum este definit în Secțiunea 11.1. Folosind noțiunea ecuației (11.12) aflăm că dacă *x^t^xi^* atunci următoarea ecuație trebuie să fie valabilă

^^(rp + or-i/Vi-i^aP + • • • + *a* $_{l}$ -„*N* $_{l_\ mi}$ „*(j;) = yj*

\<*j^n* (11.20)

Vor exista *n* astfel de ecuații cu *k+m* necunoscute. Fiecare dintre ele are doar m + 1 termeni, astfel încât matricea rezultată este bandă cu cel mult *m* diagonale inferioare și m superioare. (Comparați și forma exemplului ILL) Forma exactă depinde de pozițiile relative ale punctelor de întrerupere și ale punctelor de date. Când punctele de întrerupere sunt distribuite uniform și, de asemenea, coincid cu ordonatele punctelor de date, forma este deosebit de simplă. În fiecare punct avem doar m B-spline nenule și valorile lor sunt determinate din ecuațiile (11.10) sau (11.11). Pentru cazul patratic avem

a, + 4^_| — 2/, i — 1, 2, • ■ • 4 . (11.21a)

Pentru cubic

a $_{t}$ + 4a,_| + a $_{(_ 2}$ — 6/ $_{f}$ r — 1, 2, • • • n . (11.21b)

Condițiile de la punctele finale trebuie adăugate acestor sisteme de ecuații. În caz contrar, soluția lor este simplă. Dacă se presupune că curba este periodică, atunci în loc de condiții suplimentare de final, reducem numărul de necunoscute prin stabilirea unui $_0$ egal cu a„, etc.

Cazul general este ceva mai complicat. În mod clar, nu ar trebui să existe mai mult de m+1 puncte de date pe fiecare interval, altfel sistemul ar putea fi supradeterminat. Dacă punctele de date alternează cu punctele de întrerupere, atunci pot exista cât mai puține m+1 diagonale nenule. Relația corectă între punctele de

întrerupere și punctele de date este importantă, iar o caracterizare este dată de următoarea teoremă.

Teorema 11.2: Problema de interpolare spline are o soluție unică dacă și numai dacă

^(0)^0. /-1,2. ■« (UM)

Omitem demonstrarea acestei teoreme (vezi [ILSW]) și subliniem doar interpretarea acesteia. Pe fiecare subinterval dintre punctele de întrerupere există doar m+1 B-spline nenule și, prin urmare, pentru a satisface condiția teoremei, subintervalul nu poate conține mai mult de m + 1 puncte de date. Acest lucru asigură că sistemul de ecuații (11.20) nu va fi supradeterminat. De asemenea, știm că B-spline *N^* este nenulă numai peste subinterva) (x)^+ JH + 1) și, prin urmare, acesta este singurul subinterval în care se poate afla punctul de date *tj* . Deoarece numărul de spline B din ecuația (11.12) este egal cu numărul de puncte de date (n -A+m), avem câteva constrângeri suplimentare. Prima dintre liniile B este diferită de zero numai peste primul subinterval și, prin urmare, acel subinterval trebuie să conțină un punct de date. Același lucru este valabil și pentru ultimul subinterval. Dacă primul subintcrval conține două puncte, al doilea poate să nu conțină niciunul. În caz contrar, trebuie să aibă cel puțin unul și așa mai departe.

Algoritmul ll.l Interpolare cu B-spline.

Notație: (t^Y \<j^n **este mulțimea punctelor dau, x (, lS/sA+l este mulțimea punctelor de întrerupere, m este gradul splinei, iar** *r* **este o variabilă auxiliară.** *B(JJ)* **este matricea sistemului de ecuații (11.20)** *WHN.^Y*

0. **Dacă** există un *r pozitiv* astfel încât x m _,_,<r ,<;x w _„ **atunci** fă din x 0 un punct de întrerupere de multiplicitate m-r+l. Dacă există un *r pozitiv* astfel încât *Xk-„+r ^t K ^Xi-^+f+t,* **atunci** fă din x 4 un punct de întrerupere de multiplicitate r+2.
1. Pentru /—l să « verifice că *Xj-^ <tj<xj.*
2. Pentru i=l la *k* verificați că nu există mai mult de m+1 puncte de date pe intervalul (x,-^).
3. **Pentru y —** 1 la **n faceți:**

   ÎNCEPE.
4. Găsiți cel mai mare *i* astfel încât *x^tj.* {Acest lucru implică că 0^*i)
5. **Pentru** / - i la *i—m* prin — eu **fac:**

   ÎNCEPE.
6. Procedura de apel *BSPLINE(i ,tj ,m).*
7. Set *B(j,i)* egal cu *N/^U,).*

   Sfârşit.

   Sfârşit.
8. Utilizați o procedură standard pentru rezolvarea ecuațiilor liniare cu matrici cu benzi pentru a găsi a_w , o_ w+b -- o 4 _, din *B(j,I)* și *y^yi. - ■ y^*

9. **Sfârșitul algoritmului.**

Algoritmul 11.1 verifică condițiile teoremei, stabilește sistemul de ecuații (11.20) și apoi folosește un program de bibliotecă pentru a le rezolva. Pasul 0 introduce mai multe noduri la punctele finale. Acest lucru este necesar pentru a avea m+1 B-spline pe fiecare interval în care există un punct de date. Se pot seta oricând multiplicitățile lui x_0 și x* la m+1 fără a efectua verificările. În pasul 4 al algoritmului, unii dintre indicii de matrice pot fi negativi. Dacă implementarea este într-un limbaj de computer în care indicii negativi sunt ilegali, atunci se poate modifica algoritmul adăugând *m* la valorile subscriptelor matricei.

Exemplul 11.2: Găsiți o spline de interpolare de ordinul întâi pe intervalul [0,7] cu puncte de întrerupere la x—2 și x—5 pentru punctele de date (1,1), (3.1). (4,2) și (6,0). În acest caz, ambele puncte finale trebuie să fie duble. Este ușor de verificat că spline de ordinul întâi de interpolare constă din linia care unește (3,1) și (4,2) peste (2.5). linia de unire (1,1) și (2.0) peste [0,2] și linia de unire (5,3) și (6,0) peste (5,7) (vezi Figura 11.5).

Derivarea formală a acestui rezultat este lăsată ca exercițiu (vezi problema 11.6). □

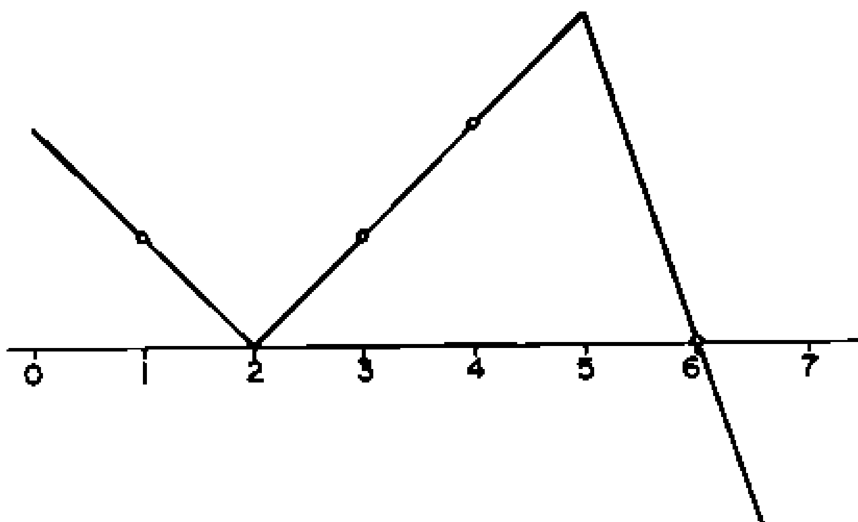


Figura 11.5 **O spline de interpolare liniară cu două puncte de întrerupere și care trece prin patru puncte. Relația dintre punctul de date și locația punctului de întrerupere ilustrează Teorema 11.2.**

11.6. B-SPLINES ÎN GRAFICE

Este posibil să se utilizeze B-spline pentru a genera curbe într-un mod similar cu polinoamele Bezier. Dacă P, (f — 0, 1, • • • A) este un set de puncte de ghidare, atunci definim o spline ca

P(O - £ A/Wr) (H23) 1-0

Domeniul lui *l* nu mai trebuie să fie [0,1], dar trebuie să definim punctele de rupere f $_{H}$ fj, • • • f $_{4}$ _ $_{h}$ ceva care nu era necesar în cazul polinoamelor Bezier. Rețineți că ecuația (11.23) este echivalentă cu două perechi de ecuație (11.12) și punctele de ghidare corespund coeficienților a $_{(}$ ai acelei ecuații. Observăm că P(f) este o sumă de vectori înmulțită cu numere (B-splines) care însuși se adună la unu datorită teoremei 11.1. Prin urmare, avem următorul rezultat.

Propozitia 11.2: *P(j)* se afla in carcasa convexa a cel mult m+l punctelor de ghidare *P,.* □

Pentru w - 1 avem poligonul definit de punctele de ghidare. (Fiecare *P(t)* se află pe linia care unește două puncte.) Triunghiurile umbrite din Figura 11.6a arată partea planului în care se află curba *P(t)* pentru m — 3 în două exemple. Punctele multiple forțează curba să treacă mai aproape de punctele de ghidare, aproape în același mod ca și în cazul polinoamelor Bezier. Figura 11.6b arată că, pentru m — 2. puncte duble

forțează curba să treacă prin ele, h este posibil să se obțină limite și mai strânse pentru locația unei spline (a se vedea Exemplul 11.4 de mai jos), dar o comparație între regiunile umbrite din Figura 11.6 și carcasa convexă a tuturor punctelor (oferă liniile de curbe mai strânse) decât fac polinoamele Bezier.

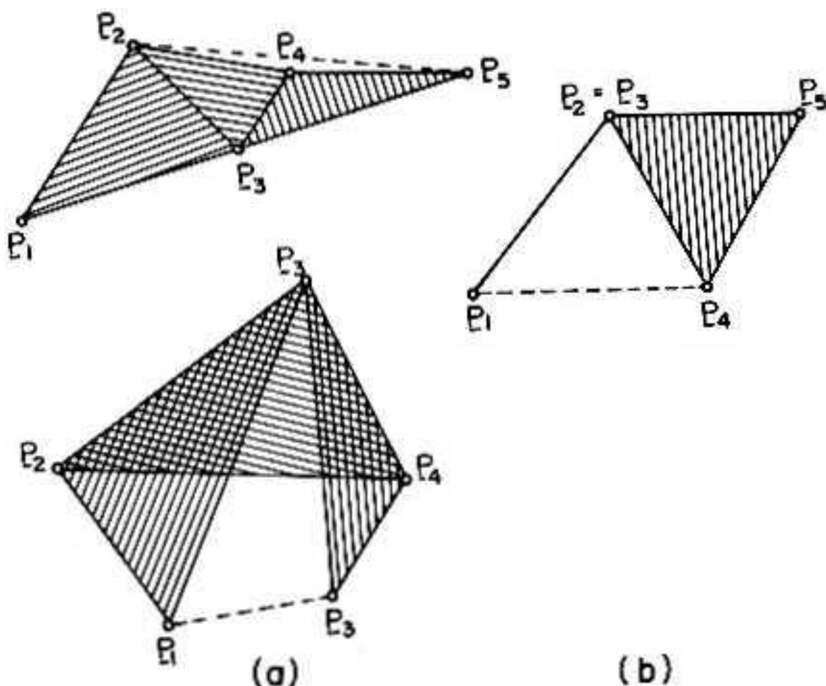


Figura 11-6 Părțile planului în care trebuie să se afle o spline pătratică cu punctele de ghidare date sunt umbrite. Liniile punctate immit partea în care trebuie să se afle un polinom Bezier cu aceleași puncte de ghidare.

Formularea ecuației (11.23) are avantajul că *P* poate fi un vector tridimensional, astfel încât se pot produce curbe de spațiu în acest fel.

Ecuația (11.23) poate fi interpretată într-un alt mod dacă ne gândim la *P(t)* și la ^s ca numere complexe. Este pur și simplu o spline complexă

Hr) „S^^(O (U.239
*-0

Deoarece numerele complexe au o interpretare geometrică (cu x corespunzând părții reale și *y* imaginarului), cele două forme sunt echivalente. Formalismul numerelor complexe a fost folosit explicit de Knuth în designul fonturilor (vezi Notele bibliografice).

Ne vom uita acum mai atent la comportamentul spline-urilor exprimate prin B-splines. Fie *B^j(t)* valorile lui *N^lf}* pentru *t t +j<t <t t +j+i.* Valorile lui *j* care prezintă interes sunt 0. 1, • ■ ■ *m.* Apoi. Ecuația (11.23) poate fi scrisă ca

'(OS *Pt-jB,* 0) *t^mi^.* (11.24) *jo*

Dacă există numai * + 1 puncte P o , *P lt* • • • *P k ,* atunci spline poate fi evaluat doar pentru *t >l„* . În general, F(r w)#^ 0 și ^(l)*^*- Pentru a ne asigura că spline trece prin punctele de capăt, trebuie să le facem multiple. Este întotdeauna adevărat că, dacă un punct are multiplicitatea *m,* argumentul poate fi, de asemenea, mai degrabă să treacă prin el. văzut din ecuaţia (11.24) Dacă ^-i - *P,^ 2* = ■ • ■ — *P t -„* avem

P(O -A-i *ZB.-^)* ★*W^')* r, </</,+,. (11.25)

Din cauza Teoremei II.I termenul dintre paranteze este egal cu •”^^.,o(0- Din ecuaţia (11.5) vedem de asemenea că S^.oO) este proporţional cu (rG)". Dacă *c* este o constantă adecvată avem

/»(DA-JJ -^(f“ffr] +A^a-^r • (H.26)

Pentru *t ^t f* se obţine *P(t,) = P,^.* De asemenea, pentru o«/£r /+1 , spline se află pe linia care unește vectorii *P^ }* și *P t .*

O mai bună înțelegere a comportamentului spline-urilor poate fi obținută prin luarea în considerare a unor cazuri mai simple. Dacă punctele de întrerupere sunt distribuite uniform, atunci putem folosi expresiile ecuațiilor (11.9) la (11.11) și variabila normalizată w. Observăm din ecuația (11.8) că definiția lui u depinde de *i* și o anumită grijă este

necesar atunci când amestecăm expresii pentru B-spline pe intervale diferite. Cu toate acestea, este ușor de arătat că dacă înlocuim u cu u+J în expresiile pentru *U t -j^,* atunci formulele devin consistente cu cele pentru *U,^*. În acest fel, ecuația (11.24) ia următoarele forme

mk P(£(/+u))-uP,+(1-«)^-, (11.27a)

m =• 2: P(£G+u)) + ^*P,* + y - [u - y)^2 *P,*
2

$$+\frac{1}{2}(u-1)^2P_{i-2}.$$ (11.27b)

m — 3: J'(L(/+u))+^^{J}F$_/$ + ^ 1_±£Z11 (*-l)^2^
3 2

f^ 6 (11.27c)

In all cases Oiuil. These expressions can be rewritten powers of u as as sums of follows.

m 1: P(£(J+U)) -(P, -P,-.)« + P/-i (11.28a)

m 2: **P(£O+u)) ≡**

$$\left[\frac{P_i+P_{i-2}}{2}-P_{i-1}\right]u^2 + \left[P_{i-1}-P_{i-2}\right]u + \frac{1}{2}\left[P_{i-1}+P_{i-2}\right].$$ (11.28b)

m = 3:

$$\equiv \frac{1}{2}\left[\frac{1}{3}P_i - P_{i-1} + P_{i-2} - \frac{1}{3}P_{i-3}\right]u^3 + \left[\frac{1}{2}P_{i-1} - P_{i-2} + \frac{1}{2}P_{i-3}\right]u^2 + \frac{1}{2}\left[P_{i-1} - P_{i-3}\right]u + \frac{1}{3}\left[\ldots\right]$$ (11.28c)

1^P1+iP.-^P,-, =

Expresiile de mai sus sunt convenabile pentru studiul diferitelor proprietăți spline. În special, putem obține omologii ecuației (11.26) pentru m - 2 și m - 3. (Ecuația (11.27a) este omologul său pentru m = 1.) Obținem

m-2: P(L(i+w)) $= \frac{1}{2}\left[P_i - P_{i-1}\right]u + P_{i-1}$ (11.29a)

m - 3: *P(L(i+u))* $= \frac{1}{6}\left[P_i - P_{i-1}\right]u^3 + P_{i-1}$

(li.29b)

În toate aceste cazuri P(Li) - P,^ Pe de altă parte

P«i+1)£) »y^+Pa forma -2. (11.30a)

şi

P((i+1)L)--T-Pj-rf—P, forma -3. (11.30b)

Dacă m+l punctelor de ghidare se află pe o linie dreaptă, atunci Propoziția 11.1 arată că spline-ul se va afla și pe aceeași linie. Ecuația (11.24) spune că acest lucru se va întâmpla pentru/, ^r <r f+ ,.

Este posibil să se producă curbe închise prin definirea P_, = *P„, P-2* = ^-i. etc. și, de asemenea, r_, = *t n*, r_ 2 = /„_, etc.

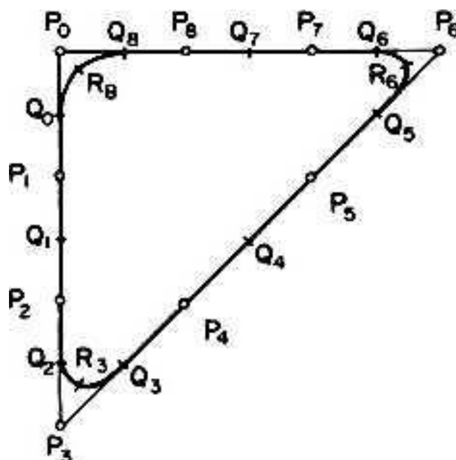


Figura 11.7 Potrivirea unei spline pătratice la un triunghi

Exemplul 11.3: (a) Luați în considerare cele nouă puncte din Figura 11.7 unde ^o* ^1'^2» Fj), (P 3 , *P 4 , P^, p* 6) și (P*, *P 7 , P^PQ)* sunt mulțimi de puncte colliniare. Pentru a potrivi o spline pătratică uniformă, folosim ecuația (11.30a) pentru a identifica punctele 20.01. • '08 unde *Q t ~ P(iL).* Setarea « = 1/2 în ecuația (11,27b) găsim

? (0 + y)t) = jJ' (- 2 + yP / - 1 +yP / **(11.31a)**

sau

$$P(«+|)£)-|^. \quad i-2 + \frac{3}{4}P_{i-1}\Big] + \frac{1}{2}\Big[\frac{3}{4}P_{i-1} + \frac{1}{4}P_i\Big] \quad (11.31b)$$

A doua ecuație sugerează o construcție grafică pentru găsirea punctelor *R 3, R^* și *R %* din figura 11.7. Desigur, ori de câte ori cele trei puncte sunt coliniare, P((i+l/2)£) coincide cu PH .

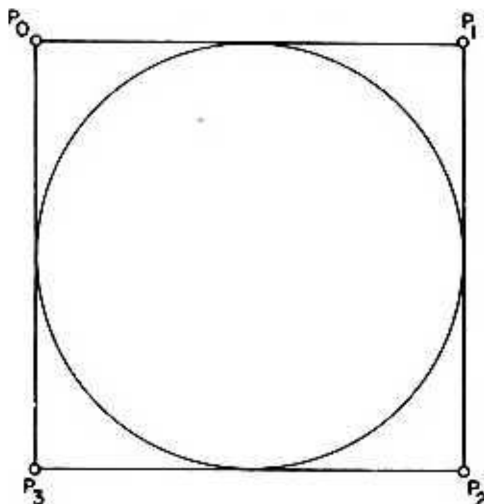


Figura 11.8 **Potrivirea unei spline pătratice la un pătrat**

(b) Figura 11.8 prezintă o spline periodică ghidată de patru puncte, formând un pătrat. Deși arată ca un cerc, nu este. Raportul dintre razele diagonalei și razele verticale și orizontale este 3/(2V2) - 1,06. □

Ecuația (11.24) arată că, spre deosebire de spline de interpolare, spline ghidate arată într-adevăr o dependență locală de cel mult *m* puncte.

11.7. DESCRIEREA FORMEI ȘI B-SPLINES

Există anumite aplicații în care este important să descrieți forma unei curbe într-un mod foarte precis. Proiectarea matematică a fonturilor de caractere pentru fotocompozitoare controlate de computer este o astfel de aplicație. În anumite mașini comerciale caracterele sunt definite printr-o codificare vectorială a contururilor lor. Contururile sunt apoi completate de un algoritm de tipul discutat în Secțiunea 8.2. Utilizarea vectorilor liniari necesită specificarea unui număr foarte mare dintre ei pentru a crea impresia de netezime. (Acest text a fost produs printr-un astfel de sistem). Potrivirea curbei cu spline oferă câteva posibilități interesante, deoarece poate fi necesar să specificați doar punctele de ghidare, mai degrabă punctele de contur efective. Următorul exemplu oferă câteva formule utile pentru un astfel de efort.

Exemplul 11.4: Investigăm forma splinei pătratice cu distribuție uniformă a punctelor de întrerupere. Ecuația (11.28b) arată că *P(iL)* este întotdeauna dată de ecuația (11.30a), adică spline trece prin punctele medii ale laturilor poligonului format de punctele de ghidare. Prin diferențierea ecuației (11.27b) rezultă

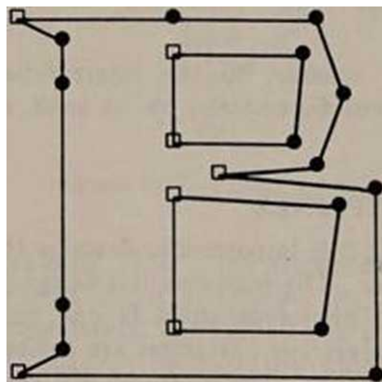
= [p.+^-j-ZP,-! *συ* L

Înlocuirea u=0 sau u = I în ecuația de mai sus]u + [PJ-.+P^J] (11.32) arată că spline este tangentă la laturile poligonului punctelor de ghidare pentru aceste valori ale lui w. Dacă rearanjam termenii ecuației (11.27b) obținem următoarea expresie:

P(u)-y[p / 4^-.,]u 2 + 2(uu 2)P i _ 1 + -i-[p (_^^ (11,33)

Se poate verifica că suma $u^2 + 2(u—w^2) + (w—I)^2$ este egală cu unu, astfel încât spline se află în triunghiul format dintr-un vârf al poligonului de ghidare și punctele medii ale celor două laturi adiacente. □

Restricția privind locația spline derivată în exemplu este semnificativ mai strictă decât cea prevăzută de Propunerea 11.2. Prin urmare, splinele pătratice cu distribuție uniformă a nodurilor sunt utile în aplicațiile în care se dorește să se reprezinte forma cu acuratețe și, în special, să se producă contururi netede folosind foarte puține puncte de date. Figura 11.9 prezintă un aranjament de puncte de ghidare și spline rezultată formând conturul unei litere. Fontul este specificat de douăsprezece puncte pentru conturul exterior și patru pentru fiecare contur interior. În acest sens, ele sunt superioare spline-urilor cubice în care relația dintre poligonul de ghidare și curbă nu este la fel de strânsă. Teoretic , o spline pătratică necesită mai multe noduri decât o spline cubică pentru a aproxima o curbă dată cu același grad de precizie. Cu toate acestea, creșterea numărului de noduri pare a fi nesemnificativă în multe probleme practice. Cititorii se pot convinge cu privire la aceste puncte încercând să realizeze conturul literei „B” prezentat în figura 11.9 (sau un contur similar) printr-o spline cubică. O spline cubică este mai convenabil de utilizat în aplicațiile în care o formă este definită de o secvență de puncte și tangenta la curba la acestea. Apoi arcurile dintre noduri pot fi găsite cu ușurință printr-o expresie similară cu ecuația (10.5).



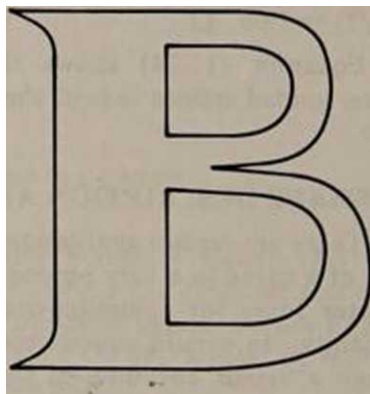


Figure 11.9 The use of guided splines with a B-spline basis for font design: the guiding polygon is shown to the left with solid circles representing the location of the simple guiding points and hollow squares representing the location of the double guiding points. The resulting spline is shown to the right.

11.8. NOTE BIBLIOGRAFICE

Spline-urile sunt tratate, cel puțin pe scurt, în majoritatea textelor moderne de analiză numerică și mai pe larg în majoritatea cărților despre teoria aproximării. Al doilea volum al cărții lui Rice [10.RI] consacră un capitol subiectului, cu accent major pe teorie. Există, de asemenea, destul de multe volume care se ocupă exclusiv de spline. Unele dintre ele oferă un tratament matematic general fără o atenție specială problemelor de potrivire a curbelor ([ll.ANW], (11.SC], (ll.SCH]), în timp ce altele se concentrează pe ultimul subiect [ll.SP]. [ll.LCS] și (ILLS] sunt colecții de lucrări relativ recente, inclusiv multe despre spline. (11.DB2]), iar cartea sa [11.DB3] este cea mai bună referință despre spline pentru oricine este interesat să le folosească. Conținutul [11.DB3] variază de la rezultate teoretice de bază despre spline la liste de programe de calculator și exemple specifice.

Riesenfeld a fost primul care a prezentat un studiu sistematic [11.RI] al utilizării B-spline-urilor în aplicațiile grafice și a subliniat avantajele acestora . [ll.BR] conține multe lucrări despre utilizarea spline-urilor în grafică și subiecte conexe. [ll.CLR] este o referință excelentă care descrie

algoritmi pentru utilizarea B-splines pentru descrierea formei. În special, ei prezintă un algoritm pentru calcularea unei spline folosind subdiviziune similară cu cea discutată pentru polinoamele Bezier în Secțiunea 10.5. Lucrarea conține, de asemenea, numeroase citări ale literaturii de specialitate.

Knuth (11.KN) a folosit spline cubice pentru designul fonturilor. El are specificat un *meiafoni* unde fiecare caracter este definit de o spline cubică cu coeficienți complexi. Caracterele reale sunt generate ca uniuni de discuri circulare sau alte forme simple ale căror centre sunt situate pe punctele unor astfel de spline.

11.9. LITERATURA RELEVANTA

Ill.ANW) Ahlberg, J. H; Nilson. RO; și Walsh. JL *„Hit Theory of Splines și aplicațiile lor".* **New York: Academic Press, 1967.**

[11.BR1 **Barnhill, RE și Riesenfeld, RF (eds)** *Proiectare geometrică asistată de computer.* **New York: Academic Press, 1974.**

U LCD **Cline, AK „Fitting cu curbe cu valori scalare și plane folosind spline sub tensiune".** *CACM.* **17 (1974) p. 218-220.**

Ill.CLRl **Cohen, E.; Lyche, T; şi Riesenfeld, R. „Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics",** *CGIP,* **14 (1980), pp. 87-111.**

(ILDBl) **de Boor. C. „Splines as Linear Combinations of B-splines", în [1 LLCS], pp. 1-47.**

[II.DB2] **de Boor, C. „Package for Calculating with B-splines",** *SIAM J. Numer Anal..* **14 (1977) pp. 441-472.**

IILDB31 **de Boor. C.** *Un ghid practic pentru spline.* **Heidelberg: Springer Ver lag. 1978.**

(II.KNI **Knuth, DE** *TEX și METAFONT: New Directions in Typesetting,* **Bedford, Mass.: American Mathematical Society and Digital Press, 1979.**

[I SRLS) **Lorentz, GG; Chui. CK; și Schumaker, LL (eds). Teoria** *aproximării H.* **New York: Academic Press. 1976.**

[ILLS) **Law, AG și Sahney, BN (eds).** *Teoria aproximării cu aplicații.* **New York. Presa Academică. 1976.**

IILRI) **Riesenfeld. R. „Aplicații ale aproximării B-spline la problemele geometrice ale proiectării asistate de computer".** *Teză de doctorat.* **Departamentul de Informatică. Univ, din Utah. Salt Lake City. Martie 1973 (Raport tehnic numărul UTEC-CSc-73-126.)**

|l ISC) **Schultz, MH.** *Späne Analysis,* **Englewood Cliffs, NJ.: Prentice Hall. 1973.**

Ill.SCH] Schumaker. LL *Spline Functions: Basic Theory,* **New York: J Wiley. 1981.**

(M.SPl Spath. H.. *Spline Algorithms for Curves and Surfaces,* **Winnipeg: Uni- us Maihematica**

Pub!.. 1974. (Trad. din germană de WD Hoskins și HW Sager.)

(11.SW1 Schoenberg. IJ și Whitney. A. „On Polya frequency functions IH,” *Trans. Amer. Mathem. Soc.* **74 (1953), pp. 246-259.**

11.10. PROBLEME

11.1. Este curba produsă în problema 10.1 o spline simplă?

11.2. Luați în considerare următoarea spline liniară cu doar un punct de întrerupere

$$p(l) = al - a(tt_b{}^\wedge$$

Calculați valoarea acestuia pentru a - IO ^{5} și t — 10 ^{N} când t_b — 2. folosind aritmetică cu un întreg întreg de precizie și presupunând că aveți un computer cu cuvinte de 16 biți. Contează dacă utilizați expresia definitorie de mai sus sau forma folosită în ecuația (11.1)?

11.3. Găsiți o expresie sau un număr pentru valoarea maximă și valorile la noduri pentru fiecare dintre B-spline-urile date în ecuațiile (11.6), (11.7), (11.9), (11.10) și (11.11).

H.4. Repetați analiza exemplului 11.1 introducând un punct de întrerupere suplimentar în intervalul (2,4).

11.5. Scrieți un program care implementează algoritmul 11.1. Aveți grijă să selectați o rutină pentru rezolvarea sistemului de ecuații care profită de faptul că matricea sistemului este în bandă.

11.6. Elaborați în detaliu Exemplul 11.2 și demonstrați în mod formal că curba dată în Figura 11.5 este într-adevăr spline de interpolare corectă.

11.7. Exprimați ecuația (11.239 >n coordonate polare. Puteți produce o spline care este un cerc prin alegerea corectă a coeficienților, punctelor de întrerupere și gradului splincilor B? Dacă credeți că acest lucru este imposibil, furnizați o dovadă formală a afirmației dvs.

11.8. Găsiți o ecuație de forma $f(x^\wedge)$ - 0 echivalentă cu ecuația (11.28b). Folosiți această expresie pentru a arăta că curba rezultată este o parabolă. *(El:* Dacă o curbă este dată de o ecuație de forma $ax^2 + 2bxy + cy^2 + dx + ey + f - 0.$ atunci este o parabolă dacă b^2 — $ac.)$

11.9. Scrieți un program pentru calcularea ecuației (11.28b) și utilizați-l pentru a obține afișaje de curbe netede definite de puncte de ghidare.

11.13. Figura 11.10 prezintă patru puncte de ghidare pentru un arc al unei spline cubice. Fie *A* să fie *P t -i, B* pentru *Pj-2, C* pentru P (_ b și *D* pentru *P {.* Punctele *E* și *F* împărțiți segmentul *AB* în trei părți egale. *G* și *H* faceți același lucru pentru *BC.* și *J* și *K* pentru *CD. L* este punctul de mijloc al *lui FG* și *M al lui HJ.* Arată următoarele legături de proprietăți pentru arcul spline pe interval (/£,(/+!)£). (a) Începe din punctul *L* și este tangentă la dreapta *FG* acolo, (b) Se termină în punctul *M* și este tangentă la dreapta *HJ* acolo, (c) Are punct de inflexiune dacă vectorii *FG* și *HJ* formează un unghi mai mare de 90*.

11.14. Modificați ecuația (10.5) astfel încât să poată fi utilizată pentru a calcula o spline cubică care trece printr-un set de puncte din plan și să fie tangentă la liniile date prin puncte.

Capitolul 12

APROPIERE AL CURBURILOR

12.1 INTRODUCERE

Ultimele două capitole s-au ocupat în principal de interpolare în care o curbă trebuie să treacă prin toate punctele de date. (Punctele de ghidare fac parte din parametrii de proiectare, astfel încât o curbă definită de aceștia poate fi folosită în continuare pentru interpolare.) Există multe aplicații în care interpolarea nu este necesară, sau chiar de dorit, și se dorește ca curba să treacă numai lângă punctele de date. Aceasta este problema de aproximare. Dacă potrivirea curbei se face într-un mod interactiv, distincția dintre cele două probleme nu este esențială. Utilizatorul modifică parametrii (cum ar fi punctele de ghidare) până când curba arată corect. „Priviți la dreapta" poate însemna că curba trece prin toate punctele de date, sau prin majoritatea dintre ele, sau aproape de toate și așa mai departe. Dacă potrivirea curbei se face automat , astfel de criterii subiective trebuie înlocuite cu măsuri precise din punct de vedere matematic de apropiere. Cele mai comune astfel de măsuri sunt eroarea maximă și eroarea pătrată integrală (ISE). Eroarea poate fi măsurată fie de-a lungul unei coordonate, fie de-a lungul unei normale la curba de aproximare. Acesta din urmă este intuitiv mai atrăgător, dar mai dificil de calculat. Fie e_t eroarea punctuală în punctul i'*, adică distanța dintre curbă și punct (măsurată de oricare dintre tehnicile de mai sus). Atunci avem

Eroare maximă ■ E_{MK} '' maxlej (12. la)

Eroare pătrată integrală ■ $E^{\wedge}$ — J ^{e} ^ (12.lb)

Deoarece aproximările ISE sunt mai manevrabile din punct de vedere matematic decât alte forme, acestea sunt mai populare. Se pot găsi formule închise care sunt valabile nu numai pentru aproximări prin polinoame, ci și pentru aproximări prin spline și alte curbe. Secțiunile 12.2 și 12.3 prezintă proprietățile fundamentale ale unor astfel de aproximări atunci când curbele de aproximare sunt fie polinoame, fie spline cu noduri fixe. Pentru a găsi aproximările maxime ale erorilor, trebuie să rezolvați o problemă de programare liniară sau să utilizați un alt algoritm iterativ. Găsirea aproximării optime poate necesita astfel un calcul semnificativ . făcând algoritmii euristici suboptimali mai populari. Din punct de vedere practic, cea mai importantă problemă este una care este practic insolubilă din punct de vedere matematic: găsirea unei aproximări printr-o spline cu noduri variabile. Secțiunile 12.4 și 12.5 discută unele aspecte ale acestei probleme, dar un tratament complet este în afara domeniului de aplicare al acestui text (vezi Notele bibliografice). Secțiunea 12.6 prezintă câteva aplicații ale potrivirii curbei în grafică.

12.2 EROARE PĂTRAT INTEGRAL APROXIMAREA

Fie definită o curbă de aproximare ca

$$*(O" 2 \, a_j b_j(t) \quad 7\text{-}0 \qquad (12,2)$$

unde $b_j(t)$, j = 0, 1, • * • m este o familie de curbe care formează o bază. În termeni simpli, aceasta înseamnă că toate curbele netede de interes pot fi exprimate într-un mod unic prin ecuația (12.2). Cerința de unicitate are următoarele implicații:

Propoziţia 12.1: Fie MO, ■ ■ MO o bază. **Dacă** există coeficienţi a_Q ' • a_n, astfel încât

$$\sum^{m} a_j b_j(t) - 0 \text{ pentru tot } t, \qquad (12.3)$$

apoi o_0 - a l • • • - o_m = 0. Este adevărat și invers. Fie F o mulțime de m funcții care pot satisface ecuația (12.3) numai având toți coeficienții a_t zero. Atunci, dacă o altă funcție poate fi exprimată ca o sumă a ^(r), expresia respectivă este unică.

Dovada: Să presupunem că există coeficienți nenuli pentru care ecuația (12.3) este valabilă. Lasă

*(O = 2 a/b^t) (12,4)
>-o

unde A(c) este o funcție arbitrară. Adunând ecuațiile (12.3) și (12.4) aflăm că

MO-S <^+a/)M') (12,5)

adică *h(t*) poate fi exprimat ca o sumă a funcțiilor de bază în mai multe moduri, ceea ce este imposibil. Astfel, propoziția este dovedită prin contradicție . Purtarea acestui argument în direcția opusă dovedește inversul propoziției. □

Fie acum (x (^) / = 0, 1, 2, • • • n punctele date date. Pentru a studia problema de aproximare, este convenabil să definim un parametru *t* și să asociem valorile lui *I* cu fiecare punct, astfel încât să avem triplete *(t,^, ,yi).* Două stiluri de bază pentru o astfel de atribuire sunt:

(a) *Forme de undă:* Dacă succesiunea x/s este în creștere, putem seta

x "

și considerați datele ca o formă de undă în intervalul [0,1). Atunci avem

?i = K ~^U)“^“2 a j b j <6) (12-5a)
jo

şi

E 2 = 2 ^ 2 = 2 ^~2 a) b jM (12.5b)
r-0 <-0 /-0

(b) *Contururi:* O altă abordare este să setați *ti=iL,* unde L = l/n, și apoi să luați în considerare aproximații atât în x, cât și în *y* direcții pentru valorile lui *t* în intervalul [0.1],

^ = x, -g* Ui) = ^“2 a ï b jUi) (12.6a)
>-o

^ ~y~g y U t } ~y~% ^b f Ui)
/“0 (12.6b)

Se poate minimiza apoi separat

£5 - 5 <^)? and £5 1-0 - 2 W2 /-0 (12.7)

Din cauza asemănării dintre ecuațiile (12.6) și ecuația (12.5a), luăm în considerare doar cazul formei de undă. Ecuația de diferențiere (12.5b) față de *a^* găsim

du4 --2S 1-0 ^"2 a,biM /-0 *b.M* (12.8)

4 = 0. I. ■ ■ m.

Minimitate de £. necesită ca toate laturile din dreapta să fie zero, ceea ce conduce la următorul sistem de ecuații:

2 2 M0MO «/ = 2 *b,My,* **(12.9)**
***l*-0 *l*-o** ***l*-o**

A = 0, I, ■ ■ ■ m.

Se poate demonstra că pentru funcțiile pentru care Propoziția 12.1 este valabilă, sistemul de ecuații (12.9) are o soluție unică; și, cel puțin în principiu, problema de aproximare ISE poate fi rezolvată prin rezolvarea ecuațiilor (12.9). Aproximarea conturului poate fi rezolvată într-un mod similar. Rețineți că în acest caz putem defini vectori

$$P_i = \begin{bmatrix} a_i^x \\ a_i^y \end{bmatrix} \quad (12.10)$$

și obțineți punctele de ghidare ale ecuației (11.18).

În practică apar o serie de dificultăți, în principal că matricea Gram

G k j - 2 M^U) * -0, I. - m i-0 0. I. • • «(1241)

este aproape singular pentru unele dintre cele mai comune funcții de bază. În special, acesta este cazul când funcțiile de bază sunt puteri ale lui r. iar *m* depășește 2. Această proprietate poate fi văzută cu ușurință dacă înlocuim sume cu integrale, adică dacă presupunem că n—co. Apoi găsim pentru cazul puterilor lui /

Gt/ = ^'*th^a* - ^'*i^dt* ,*+/♦> ' I (12.12)
k +/+I 0 A+y+i

Valoarea delerminantului este 1/12 pentru m = I, 1/2160 pentru m = 2. și aproximativ (1/6)10 6 pentru m - 3.t

12.3 APROXIMAREA FOLOSIND B-SPLINES

Niciunul dintre rezultatele secțiunii anterioare nu depinde de structura detaliată

a funcțiilor 6 ((O. atâta timp cât formează o bază. Luați în considerare acum b-splinele de m'* grad definite pe un interval cu 4 — 1 puncte de întrerupere. Va fi un total de 4 + m — I astfel de funcții. Să presupunem că am găsit o mulțime de coeficienți a, astfel încât

$$\sum_{\bullet --\blacklozenge I}^{*} 2 «^.„(0\text{-}0 \text{ pentru toate r.} \tag{12.13}$$

Aceasta înseamnă că în orice interval suma B-spline definite va fi, de asemenea, zero. Pentru *m* - I și distribuția uniformă a punctelor de întrerupere trebuie să avem ecuații de forma

$$o\text{^-.dj)} + a,s - 0 \tag{12,14}$$

unde

$$\frac{* \sim *i}{L}$$

În mod clar, ecuația (12.14) nu poate fi o identitate în J, decât dacă ambele 4 _| și *a* t sunt zero. Acest tip de argument poate fi făcut și pentru cazul general *(m* > I și distribuția neuniformă a punctelor de întrerupere), astfel încât să se arate că B-splines formează într-adevăr o bază. În acest caz, matricea Gram va avea elemente diferite de zero numai de-a lungul diagonalelor principale și *m* diagonalelor de fiecare parte a acesteia. Într-adevăr, $N_{lm}(t)$ și *N^tf)* sunt ambele nenule într-un interval numai dacă |i-/|<m. Dacă folosim din nou integrale în loc de sume în ecuația (12.11), atunci pentru m - I și puncte de rupere uniform distanțate aflăm că primele și ultimele elemente ale diagonalei principale sunt *L/3,* cele aflate între 2£/3, iar cele de pe diagonalele de deasupra și de jos sunt *L/6.* Pentru cazul a două segmente determinantul matricei Gram este egal cu L'/I8 iar pentru trei segmente 5Z.V144 etc. (vezi problema 12.1). Inversarea unor astfel de matrici este mult mai ușoară decât inversarea matricei ecuației (12.12).

În practică, este important să se calculeze elemente ale matricei Gram pentru locații arbitrare ale punctelor de întrerupere conform ecuației (11.5). Acest lucru se poate face cu ușurință prin procedura *BSP LI NE* a

t Valoarea mică a determinantului nu este în sine un indiciu că este dificil de inversat matricea Gram, dar o discuție completă a acestui punct depășește scopul acestui text.

Algoritmul 11.la. Dacă este disponibil suficient spațiu de stocare, valorile fiecărei B-spline pot fi stocate înainte ca produsele scalare să fie calculate.

Problema critică în utilizarea B-spline-urilor sau a altor funcții polinomiale pe bucăți este alegerea locației punctelor de întrerupere. Vom discuta această întrebare în continuare.

12.4 APROXIMAREA PRIN SPLINE CU PUNCTE DE RUPE VARIABILE

Într-un sistem interactiv utilizatorul poate afișa punctele de date și apoi poate specifica locația punctelor de întrerupere printr-un editor de puncte de tipul discutat în Secțiunea 10.8. Cu toate acestea, există multe aplicații în care este de dorit să existe o selecție automată a unor astfel de puncte. Din păcate, aceasta este o problemă matematică foarte dificilă din următoarele motive. Fie T un vector care denotă locațiile punctelor de întrerupere fi, ■ ■ ■ *t r .* Înlocuim *bj{t)* cu *bj(t,T)* în ecuațiile noastre pentru a face explicită dependența bazei de punctele de întrerupere. De exemplu, cu o schimbare banală a notației, Ecuația (11.6) poate fi scrisă ca

$$M'*r)" \begin{cases} \dfrac{t-t_j}{t_{j+1}-t_j} & t_j \le t \le t_{j+1} \\ \dfrac{t_{j+2}-t}{t_{j+2}-t_{j+1}} & t_{j+1} \le t \le t_{j+2} \end{cases} \qquad (12.15)$$

Pentru a găsi aproximarea optimă atunci când locația punctului de întrerupere este o variabilă, trebuie să estimăm următoarele, în plus față de derivatele date în ecuația (12.8):

(12,16)

Pentru exemplul ecuației (12.15). găsim că

^22 f^+i
" (O +1 -M 2 o" / " // * 1 (12.17)

Ultimele două ecuații subliniază că locațiile punctelor de întrerupere intră în ecuațiile de optimizare într-un mod neliniar; nu există o soluție explicită la această problemă. În schimb, trebuie să folosiți tehnici iterative în care un set de locații este ales în mod arbitrar și apoi locațiile sunt variate pentru a reduce eroarea £ 2 - Un astfel de proces poate consuma destul de mult timp și este imposibil pentru multe aplicații.

Ca o ilustrare suplimentară a dificultăților, luați în considerare formularea ecuației (11.1) atunci când nu există constrângeri. În acest caz, dacă folosim integrale în loc de sume în ecuația (12.1b). găsim pe

următoarea expresie pentru eroare.

£ 2 = 2 *f k ^e^t)dl* (12,18)
*-sau,

unde e_k este dat de

$$/»i$$

$$MO -y. - ^a}bj{t) \qquad (12.19)$$

Dacă alegem o bază care nu depinde în mod explicit de locațiile punctelor de întrerupere (de exemplu, puterile lui /). atunci aflăm că

$$dE_2$$

$$-J—'-»'»-dacă'»)-<'t('») \qquad (12.20)$$

(Am folosit faptul că derivata unei integrale în raport cu limita sa superioară este egală cu integrandul în acel punct:

o

În mod similar, derivata sa față de limita sa inferioară este negativul integrandului din ultimul punct.) Ecuația (12.20) are o interpretare simplă: la locația optimă a punctului de întrerupere, valorile absolute ale erorilor punctwisc sunt egale de ambele părți. Dar această relație nu poate fi utilizată deoarece valorile acestor erori depind de locația punctelor de întrerupere. Dacă le introducem explicit în ecuația (12.20), atunci ne confruntăm din nou cu o ecuație neliniară.

12.5 APROXIMARI POLIGONALE

Strict vorbind, toate curbele afișate pe majoritatea sistemelor grafice sunt poligoane, deoarece sunt produse prin trasarea unor linii între pixeli adiacenți folosind o comandă precum *vec(x,y)* din Secțiunea 1.7 (Tabelul 1.2). (Dispozitivele care afișează numai puncte sunt singura excepție majoră de la această regulă.) Un rezultat matematic bine cunoscut arată că orice curbă poate fi aproximată printr-un poligon cu orice precizie dorită, astfel încât astfel de reprezentări pot avea un aspect foarte neted. Având o descriere matematică pentru o curbă C

$$f\{x,y)=d \qquad (12,21)$$

ni se poate cere să găsească un poligon care se aproximează îndeaproape cu C , având în același timp un număr cât mai mic de vârfuri. În mod clar, dacă locația

Desigur, se plătește un preț pentru confortul lui. Vezi Secțiunea 11.7 și Figura 11.9.

a vârfurilor este o variabilă, atunci putem găsi reprezentări mai eficiente . O problemă similară apare în recunoașterea modelelor, cu diferența că C este dat de un set de puncte de date mai degrabă decât de o ecuație. Potrivirea unui poligon la astfel de puncte este utilă în determinarea caracteristicilor care sunt importante pentru descrierea formei. Acest lucru este prezentat în Figura 12.1, unde unele contururi zgomotoase sunt echipate cu poligoane printr-un algoritm descris mai târziu în această secțiune.

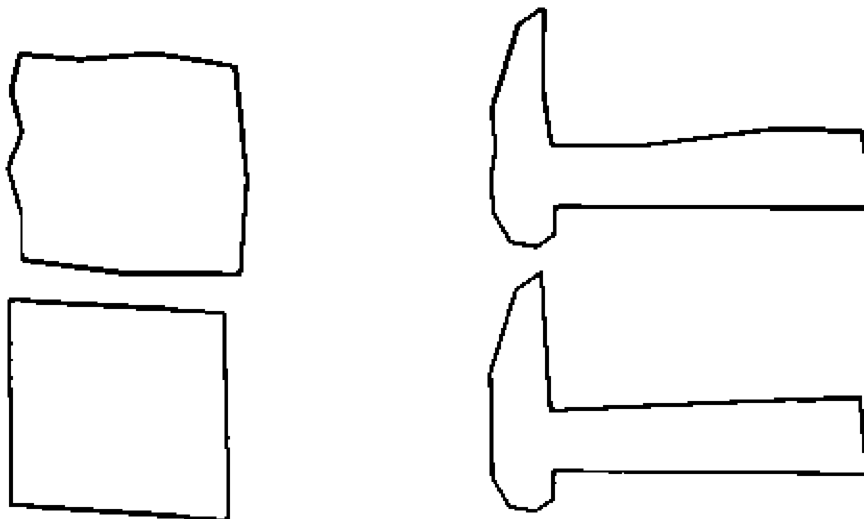


Figura 12.1 Exemple de aproximări poligonale obținute prin Algoritmul 12.1. Contururile originale sunt afișate în partea de sus și aproximările în partea de jos.

Grafica interactivă este de puțin ajutor în ambele cazuri. În primul rând, trebuie să găsim poligonul pentru a afișa curba. În al doilea, avem nevoie de funcționare complet automată. În același timp, ne confruntăm cu imposibilitatea optimizării locației nodurilor pentru spline. Pentru a găsi cel puțin o soluție suboplimală care să fie ușor de calculat simplificăm atât procedura de ajustare a liniilor, cât și procedura de localizare a punctelor de întrerupere. În special, nu vom insista să găsim numărul minim de noduri, ci doar un număr apropiat de acesta. Ideea de bază este de a examina grupuri de puncte și de a verifica dacă acestea sunt aproximativ coliniare. Dacă nu sunt, atunci grupul este împărțit până când condiția de coliniaritate este îndeplinită. Pe de altă parte, grupurile sunt

îmbinat dacă rezultatul va fi un grup cu pixeli aproximativ coliniari. Astfel, avem un proces de împărțire și îmbinare al cărui rezultat are următoarea proprietate:

Propunerea 12.2: Numărul de segmente găsite printr-o procedură de împărțire și îmbinare este întotdeauna mai mic de două ori numărul minim.

Dovada: Fie Z|. Z_2, - - / $_B$ fie intervalele partiției minime și J_x, J_2, * - • $J_„$ *intervalele* găsite prin procesul de împărțire și îmbinare. Nici un interval Z , poate conține *două* intervale J_* și $J_{k} + {}_{h}$, *altfel* lattei ar fi *fuzionat* . respectiv (sec Figura 12.2) Astfel putem avea m intervale în întregime în intervalele partiției optime, plus atâtea intervale câte puncte de împărțire, adică un total de m +m — 1. Prin urmare n < 2m — l. □.

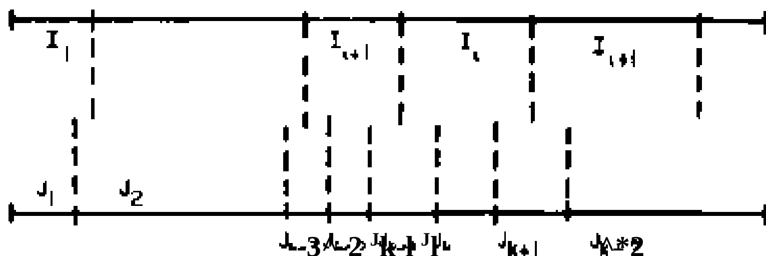


Figure 12.2 **Interval arrangement for proving Proposition 12.2**

Se dovedește că aceasta este o estimare foarte conservatoare și, în practică, n *se* află cu mult sub 2m —1. Exemplul care dă *n* = 2m - I necesită un aranjament destul de complicat între punctele de întrerupere optime și punctele finale ale grupurilor de pixeli care urmează să fie testate pentru coliniaritate. Ideea generală a unui algoritm de împărțire și îmbinare poate fi implementată în multe moduri și vom prezenta unul dintre cele mai simple în Secțiunea 12.5.2. (Consultați notele bibliografice pentru scheme mai avansate.)

12.5.1 Un algoritm de potrivire suboplimală

Orice algoritm de potrivire poligonală necesită ca punctele de date să fie subdivizate în grupuri, fiecare dintre ele urmând să fie aproximat de o latură a poligonului. Prima simplificare a problemei potrivirii poligonului este de a trage o linie între punctele finale ale fiecărui grup, mai degrabă decât de a căuta

aproximarea optimă. Linia folosită pentru aproximarea punctelor (x,^). (x,41,^1). • ■ • <x $_k$,y $_k$). este dat de

$$x(yj\text{~}yk)+y(x_k -Xj)+y_k x_r y,x_k = 0 \qquad (12,22)$$

Aceasta este într-adevăr ecuația (10.2) după înmulțirea cu numitorul, cu *y* reprezentând p $_2$ (x). *j* pentru 1. și A pentru 2. Următorul este un rezultat util.

Propoziția 12-3: Dacă un punct (uv) nu se află pe dreapta dată de ecuația (12.22), atunci distanța sa față de linie este egală cu mărimea lui *d/L* unde

$$d = u(yj-y_k) +\text{v} (x^* -x, y+ykXj-y\text{^} \qquad (12,23)$$

şi

$$L = \text{Vb^TP+Cx*-^} \qquad (12,24)$$

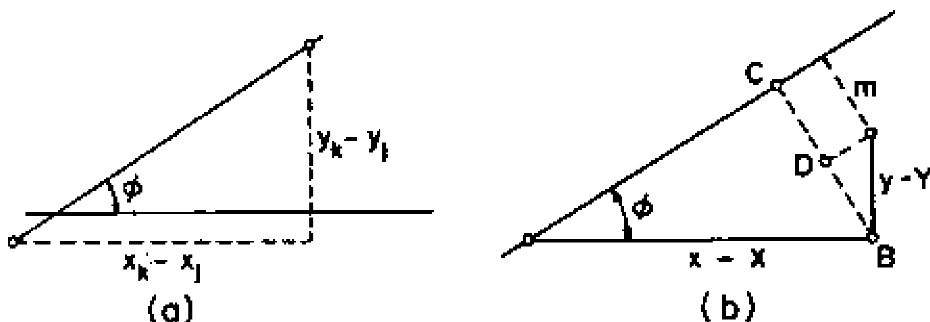


Figure 12.3 Illustration of the definitions used in Proposition I 2.3 and its proof

Demonstrație: Împărțind ecuațiile (12.23) și (12.24) termen cu termen, obținem

$$\frac{}{L} \text{" —usin0 + vcos^} + c \qquad (12.25)$$

unde unghiul 0 este definit așa cum se arată în Figura 12.3a. Constanta *c* poate fi definită explicit din ecuațiile (12.23) și (12.24), dar cel mai bine este să arăți o interpretare mai simplă a acesteia. Dacă (AW) este orice punct al dreptei, atunci stabilind *u=X* și *v = Y* în ecuația (12.25) găsim *d =0* deoarece (AW) trebuie să satisfacă ecuația (12.22). Apoi putem seta *c* = Asin0-Kcos(i> și rescriem ecuația (12.25) ca

$$\frac{d}{y} = \text{-(«-A')sin0 + (vr)cos0} \qquad (12.26)$$

Luați în considerare acum configurația din Figura 12.3b. *m* este distanța punctului de la linie. În mod clar, |BC| = (u-AJsin^ și I#/)! = (v —K)cos0. Distanța de la linie este într-adevăr

$$m = |CD| - \text{IBDI- IflCl " (v —K)cos0 —(u—A)sin^}$$

Rețineți că semnul *d/L* oferă o indicație a părții liniei în care se află punctul. □

O verificare simplă a coliniarității poate fi efectuată prin evaluarea expresiei părții stângi a ecuației (12.25) pentru toate punctele intermediare. Se poate stabili o distanță maximă astfel încât punctele unei mulțimi să nu fie coliniare dacă unele dintre ele sunt mai departe de linie decât acel maxim. Din punct de vedere practic, acest lucru nu este întotdeauna de dorit. Într-adevăr, se pot pondera erorile care par a fi sistematice diferit de erorile aleatorii. O posibilă măsură a aleatoriei este numărul de modificări de semn în *d* (vezi Figura 12.4a). Dacă în loc de un maxim absolut preferăm unul normalizat în raport cu lungimea liniei *L,* mărimea critică devine d/L^{2}.

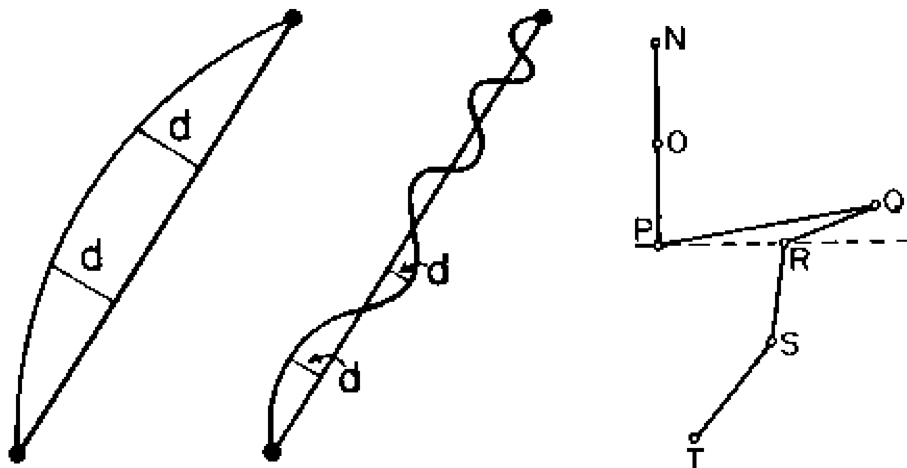


Figura 12.4 (a) Mărimea erorii *d* este măsurată de-a lungul normalei dreptei, așa cum este ilustrat în două puncte în fiecare dintre exemple . Nu există modificări de semn în primul exemplu, în timp ce există șase modificări în al doilea, (b) Un caz singular în care eroarea maximă este aproape de zero, dar punctele nu sunt coliniare dacă sunt considerate ca o mulțime ordonată.

Aceste teste dau rezultate greșite în cazuri singulare, cum ar fi cel ilustrat în Figura 12.4b. Acolo mulţimea ordonată de puncte este P. 0. şi

R și face parte dintr-un contur mai mare, așa cum se arată în celelalte puncte din figură. Punctul *Q* are o distanță foarte mică de linia care unește *P* și *R* , dar cele trei puncte nu sunt coliniare când sunt privite ca o secvență ordonată. Pentru a evita astfel de cazuri singulare, ar trebui să verificăm lungimea arcului, $L_{de\text{-}a}$ lungul secvenței de puncte și să o comparăm cu lungimea dreptei *L.* Dacă raportul dintre L_a și *L* este prea mare, atunci punctele nu pot fi coliniare. Acesta este un test simplu și este recomandabil să îl aplicați înainte de alte calcule. Experiența practică cu o varietate de date, precum și calculele cu modele geometrice simple, indică faptul că, dacă raportul L_a/L este sub 1,1, putem presupune coliniaritatea fără alte teste, în timp ce dacă depășește 1,5 putem respinge coliniaritatea complet.

Utilizarea dreptei care unește punctele extreme în locul dreptei de aproximare optimă nu are ca rezultat abateri majore de la optimitate deoarece distanța maximă a

punctelor de la interpolant nu depășește de două ori valoarea distanței de la aproximația optimă (vezi problema 12.3).

Aceste concepte sunt implementate în procedura *COLU NEAR (j ,k ,m y a .L).* listat ca algoritmul 12.1. Pasul la verifică cazul singular și poate fi omis dacă nu ne așteptăm la astfel de cazuri în date. Pasul 8 conține formulele pentru eroarea maximă normalizată și nenormalizată *T.* Algoritmul poate fi simplificat prin eliminarea pașilor 6 și 7 și a tuturor celorlalte referințe la d_Q și *C* . Apoi verificarea pasului 9 va fi o comparație între *T* (distanța maximă a unui punct de la linie) și T_Q. Cu toate acestea, rezultate mai bune sunt obținute prin utilizarea unui test pe planul *TC* , așa cum se arată în Figura 12.5. O justificare formală a abordării poate fi făcută dacă privim testul de coliniaritate ca testare a ipotezelor, dar o discuție a acestei interpretări este în afara domeniului acestui text și propunem utilizarea regiunii prezentate în Figura 12.5 ca o euristică.

Experiența practică cu astfel de algoritmi arată că criteriile care dau cele mai bune rezultate într-o aplicație nu o fac în alta. De exemplu, dacă sursa majoră de erori este cuantizarea, atunci folosirea erorii maxime fără normalizare dă rezultate mai bune decât dacă normalizăm în funcție de lungime. Pe de altă parte, dacă potrivim linii la punctele de date obținute din digitizarea unui desen cu mână liberă, găsim rezultate mai bune la normalizare. În acest caz, normalizarea pare să fie justificată deoarece oamenii trag linii lungi cu mai puțină atenție decât cele scurte.

Cititorul ar trebui să fie conștient de faptul că orice implementare a algoritmului 12.1 trebuie să fie în virgulă mobilă, mai degrabă decât în aritmetică cu numere întregi. De asemenea, ar putea fi o idee bună să împărțiți valorile a(l), a(2) și a(3) la *L* în avans, în loc să așteptați să împărțiți *T* la *L'* sau *L.*

Algoritmul 12.1 *Procedura COLLINEAR (j ,k ,m .a ,L)*

Notație: **Matricele x() și y() sunt definite global și sunt cunoscute de procedură,** *j* **este indicele primului punct și** *k* **indicele ultimului punct al grupului de puncte testate pentru coliniaritate. Tabloul** *a* **are dimensiunea trei și conține coeficienții dreptei care unește** *x(j).y(j)* **și ^(M>P(i).** *tn* **este indicele punctului în care apare eroarea maximă (în valoare absolută), adică x(m).y(»i) se află cel mai îndepărtat de linia care unește primul și ultimul punct .** **L** *este lungimea liniei* **și variabila locală .** *T* **mărimea erorii maxime (în valoare absolută),** *d* **mărimea erorii într-un punct,** *C* **numărul de modificări de semn în eroare.**

1. Evaluați: a(l) = *y(j) ~ y(k).* o(2) =x(*) -x(j).

a (3) =r(*)£<ZLz2<>) x (^ *L* = Vaa^ + ^A

la. {Opțional} Calculați lungimea totală a arcului

t. = Z VU(/)-* (ii))'+b- (Oj' 0-i>l 2 i-/+l

Dacă raportul L_a/L depășește 1,5 **atunci returnează** *fals.* **Dacă** este mai mică de 1,1 **, returnează** *adevărat.*

2. Set™ = *j, T* =0. d_0 = 0 și *C* = 0.
3. Pentru r = ;+l pentru a **face:**

ÎNCEPE.

4. Evaluați *d* = a(l)x(f) + a(2)^(0 + a(3).

5. **Dacă** Id I este mai mare decât *T,* **atunci** setați T - 1^1 și *m = i.*

6. **Dacă** semnul lui *d* este opus semnului lui d_0, **atunci** crește *C.* {Dacă d_0 este zero, nu se face nicio modificare în C.}

7. **Set** d_0 **= d.**

Sfârşit.

8. Înlocuiți *T* cu T/L^2 (sau *T/L)* și dacă *k > J+2* înlocuiți *C* cu *C/(k -j-2).* {Rețineți că dacă * - *J+2,* atunci C - 0}.

9. Dacă punctul *(CT)* se află în partea umbrită a planului din figura **12.5. apoi returnează** *adevărat.*

10. **Altfel return/a/se.**

11. **Sfârșitul algoritmului.**

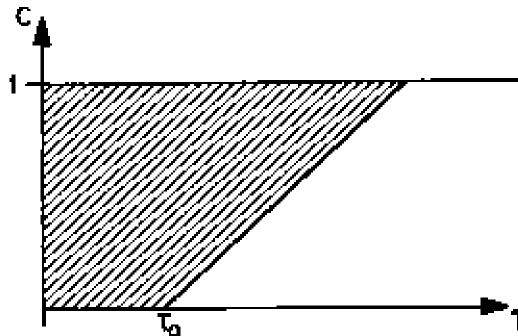


Figura 12.5 Zona umbrită este regiunea valorilor acceptabile ale *C* și *T.*

12.5.2 Un algoritm simplu de potrivire poligonală

Algoritmul 12.2 documentează potrivirea poligonului și continuă prin examinarea unor grupuri mici de puncte de dimensiunea t_0, astfel încât *k =j + k^.* (Valorile tipice ale arcului A_0 în intervalul cinci până la zece.) Ține evidența unei linii curente $L_}$ *și* a unei noi linii L_2. Dacă coliniaritatea nu va reuși pe un grup de puncte *(COLLINEAR* returnează o valoare ***adevărată*** , așa cum a fost testată în pasul 4), atunci blocul pașilor de la 5 la 10 este executat. Pasul 5 definește noua linie L_2 ca linia returnată de *COLLINEAR.* Dacă primul punct al grupului a fost stabilit ca punct de întrerupere *(p — j),* atunci noua linie devine linia curentă (pasul 6). iar pașii de la 7 la 9 au omis. Dacă *p* este diferit de *j,* atunci o verificare de îmbinare este efectuată în pasul 8 prin compararea liniilor *L* și L_2 (Figura 12.6).

Dacă unghiul dintre cele două este mic, atunci cele două grupuri sunt îmbinate și linia curentă este definită ca cea care unește *x{p),y(p)* și *x{k),y(k)* (pasul 8). Un unghi mic garantează că noua linie de aproximare. L_y, va fi aproape atât de L_t cât și de L_2 (vezi Propoziția 12.4 de mai jos). Dacă unghiul dintre linii este mare, atunci pasul 9 marchează punctul *x(j),y(j)* ca punct de întrerupere și stabilește *p = j.* În toate aceste cazuri algoritmul avansează la un nou set de puncte setând *j* egal cu *k* și incrementând *k* cu *kQ.*

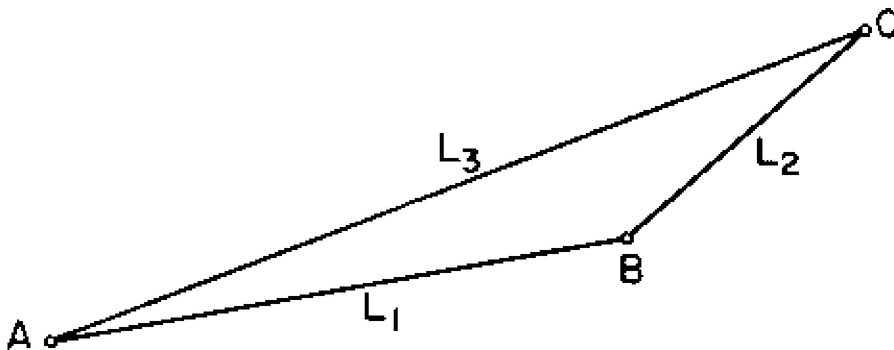


Figura 12.6 Definirea liniilor I b *L 2* . și *L*^ folosit de ritmul Algo 12.2

Dacă testul de coliniaritate al pașilor 3 și 4 eșuează, atunci testul de coliniaritate se efectuează pe grupul de puncte dintre primul și punctul în care s-a observat eroarea maximă (în valoare absolută). Acest lucru se realizează în pasul 11 prin simpla setare *k* = *m* și revenirea în partea de sus a buclei, astfel încât pasul 3 să fie efectuat din nou. (Rețineți că bucla nu crește automat *j* . Verifică doar dacă ultimul punct de date a fost atins.)

Operațiile pașilor 8 și 9 pot fi înlocuite cu o repetare a testului de coliniaritate peste unirea intervalelor printr-un apel la *COLUNEAR(p,k ,m .a ,L).* Acest lucru mărește cantitatea totală de calcul, dar garantează în mod direct că testul este valid.

Algoritmul 12.2 este strâns legat de o serie de algoritmi din literatură care „scanează" un set de puncte. În fiecare punct se reajustează linia și se verifică dacă eroarea este în limitele specificate, dacă este, algoritmul avansează la următorul punct de date, în caz contrar se introduce un punct de întrerupere (vezi Notele Bibliografice). Algoritmul prezentat aici este un algoritm „hop along". Folosește aceeași linie pentru grupuri de puncte și astfel evită efortul de recalculare a liniei în fiecare punct. Dacă apare o eroare, totuși, aceasta trebuie să revină până la punctul de eroare maximă. Astfel, este important să selectați dimensiunea grupului t_0 suficient de mică pentru a evita întoarcerea frecventă. Pentru A_0 = I algoritmul se reduce la o „scanare de-a lungul". În acest caz, testul de coliniaritate este întotdeauna adevărat și algoritmul este redus la pașii 8 și 9.

Algoritmul 12.2 Potrivire poligonală

Calion: **tablourile x() și y() conțin coordonatele punctelor de date.** *j* **este indicele primului, iar** *k* **indicele ultimului punct al grupului testat pentru coliniaritate.** A_0 **este numărul de puncte de date pe care un astfel de grup le conține în mod normal, p este indicele ultimului vârf de poligon și** L_t **este linia de la x(p)^(p) la xU)^^) "» > este indicele punctului în care eroarea maximă a fost găsită în timpul unui test de coliniaritate.**

1. Setați *p* „ 0. *j* = 0 și *k* = *k&*
2. **În timp ce** *j* este mai mic decât **n-1, faceți:**
   ÎNCEPE.
3. *Apelați COLLINEAR(jkmaL).*
4. **Dacă** valoarea returnată este *adevărată* **atunci procedați:**

ÎNCEPE.

5. Definiți linia L_2 ca fiind cea descrisă de coeficienții *a* returnați de *COLLINEAR.*

6. Dacă *p* este egal cu *j,* **atunci** definiți linia *L* l egal cu L_2.

7. **Altfel face:**

ÎNCEPE.

8. **Dacă** unghiul dintre L_t și L_2 este mic, **atunci** definiţi *L* drept linia L_3 care uneşte *x(p)y(p)* şi x(A)^(A).

9. Altfel setați *p* egal cu *j* și definiți £, ca egal cu 12-

Sfârşit.

10. Setați *j* egal cu *k* și incrementați *k* cu *k*^

Sfârşit.

11. Altfel setați A =m.

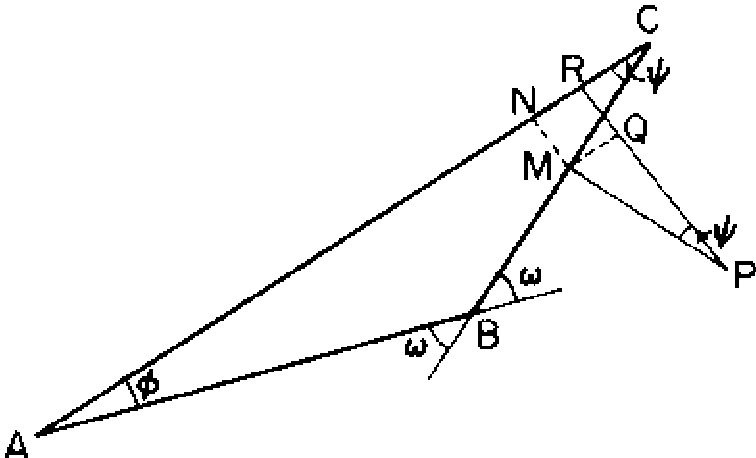
Sfârşit.

12. **Sfârșitul** algoritmului **.**

12.5.3 Proprietățile algoritmului 12.2

Următorul este un rezumat al proprietăților aproximării în anumite ipoteze despre toleranțele utilizate în procedura *COLLINEAR* și în pasul 8 al algoritmului 12.2.

Propunerea 12.4: Dacă în pasul 9 al procedurii *COLLINEAR. distanța* nenormalizată de la linie se compară numai cu un prag fix T_0 și dacă unghiul u> dintre cele două linii din pasul 8 al algoritmului 12.2 este mai mic decât sin '(TQ/L) unde *L* este maximul lungimilor lui L_2 și *Lj,* atunci distanța maximă de la L_3 va fi mai mică de $2T_0$.



Figare 12.7 Construcția utilizată pentru demonstrarea Propoziției 12.4

Dovada: Demonstrarea se bazează pe calcularea anumitor relații trigonometrice . Folosim notația din figura 12.7 și observăm că ambele linii Li și L_2 pot fi tratate în același

mod, adică L_2 poate fi fie *AB* , fie *BC* $_y$ etc. *Ly* este *AC.* Observăm că noua eroare este I/*/? Eu și noi avem

IPRl »|pei+ie«i - i/>ei+iMvi - |A/P|cos^+|Gtf|sin^ .

Deoarece *ABC* este un triunghi, unghiul ^ este mai mic sau egal cu unghiul w (w = ^ + ^. Egalitatea lui ^ și w are loc atunci când ambele sunt zero). Acolo deci

sin^ṡṡinw < -j- .

Din moment ce cos^<l avem

IPRl *£\MP* + **ICMly**

Prin definiţie *\MP* < T_o. De asemenea |CAf| < *L.* Prin urmare iPSl < T_o +*L^* = $2T_a$.

□

Limita dată de Propunerea 12.4 este mai degrabă conservatoare

din cauza întăririi inegalităţilor în timpul probei. Deși am arătat că algoritmul 12.2 dă aproximări care sunt apropiate de date (și sunt, de asemenea, continue), nu am arătat că produce un număr de segmente care este apropiat de numărul minim. Dacă presupunem că punctele de date sunt cuantizări ale unei curbe convexe, atunci inegalitățile utilizate în demonstrația Propoziției 12.4 devin mai strânse și este posibil să arătăm că numărul de vârfuri ale poligonului rezultat este aproape de minim. Cu toate acestea, deoarece aplicațiile sunt rareori limitate la curbe convexe, nu vom descrie această analiză aici.

12.6 APLICAȚII ALE APROXIMĂRII CURBEI ÎN GRAFICE

Aplicațiile de aproximare a curbelor prin poligoane sau spline cu noduri variabile sunt prea numeroase pentru a fi acoperite complet în acest text. Scurte anchete ale unora dintre ele vor ilustra relevanța subiectului pentru prelucrarea informațiilor picturale. O posibilă utilizare a editorului de puncte descris în Secțiunea 10.8 este de a introduce schițe și apoi de a cere programului să afișeze versiuni curate ale acestora. De exemplu, se poate indica o curbă, se poate informa sistemul că este intenționat să fie un cerc și curba să fie înlocuită cu un cerc aproximativ. Pentru această utilizare, editorul trebuie extins în două moduri. Una este să permită indicarea către grupuri de puncte, iar cealaltă este să includă proceduri de aproximare funcțională.

12.6.1 Gestionarea grupurilor de puncte de către un editor de puncte

Acest lucru poate fi realizat cel mai bine printr-o structură de date ierarhică. Să definim *curba structurii* după cum urmează.

curba structura — (pj.s)

unde p este un indicator către un *punct de structură* care conține primul punct al arcului. *t* este tipul structurii, iar s este o variabilă de stare. Prin tip înțelegem dacă punctele

curbei ar trebui interpretate ca puncte de ghidare ale unui polinom Bezier, sau o spline, sau puncte de date care aproximează una dintr-un set de forme de bază: cerc, pătrat, rezistor, amplificator etc. Starea indică dacă obiectul a fost șters. De asemenea, se poate extinde *punctul de structură (Secțiunea 10.8.1) pentru a include un indicator către curba* structurii căreia îi aparține fiecare punct, dar acest lucru nu este întotdeauna necesar. Într-adevăr, locul procedurii *(xyq)* din Secțiunea 10.8.2 ar putea fi aranjat astfel:

Algoritmul 12.3 *Locul procedurii (xj,q,Q)*

Notație: *x* **și y sunt coordonate date,** *q* **este un pointer către** *isapoint* **și** *Q pointer către curba* **structurii returnată de procedură.**

1. Pentru toți pointerii *Z* care indică o curbă cu *stare diferită de zero,* faceți: **Începe.**

2. Setați *2* la *Z—>p*

3. Găsiți perechea *z—>x,2—>y* care este cea mai apropiată de tox^ și fie *d* distanța dintre cele două puncte. (Secvența de puncte este căutată prin înlocuirea z cu z— >*n* și oprirea când z este zero.)

4. **Dacă** *Z* indică spre prima curbă setați £ — *Z. q =2* și *Dd.* **Altfel** comparați *d \o D.* și dacă este mai mic, înlocuiți *Q cu Z, q* cu *z* . și *D* prin *d.*
Sfârşit.

5. **Sfârșitul algoritmului.**

Dacă o comandă cu o adresă este precedată de cuvântul *obiect* (sau abrevierea lui o). apoi se aplica operatia intregii curbe. Indicatorul către curbă este valoarea *Q* returnată de o procedură precum cea enumerată mai sus. În special, primul punct al curbei este

(Q—>p)—>xAQ—>p)—>y'

12.6.2 Găsirea unor curbe simple de aproximare

Vom discuta aici câteva exemple de forme formale.

Exemplul 12.1: Găsirea celui mai bun cerc care se potrivește unui set de puncte prin minimizarea erorii pătrate integrale este echivalentă cu selectarea $x_c . y_f$. și *r* pentru a minimiza

ES ^-*J I +0'.-/r>' '4
/-I (1227)

(Acest lucru presupune că eroarea punctuală este evaluată de-a lungul normalei curbei, și nu de-a lungul unei axe de coordonate.) Minimizarea lui *E* este o problemă neliniară și se poate încerca o soluție suboptimă selectând x_f și y_f ca să fie centrul de greutate al punctelor de date. (Acest lucru este justificat numai atunci când punctele de date sunt distribuite uniform de-a lungul circumferinței .) Atunci *E* poate fi minimizat în raport cu *r.* Cea mai bună alegere este să luăm *r* egal cu raza de inerție (sec Problema 12.5). Figura 12.8 prezintă exemple de aplicare a acestei scheme suboptimale. □

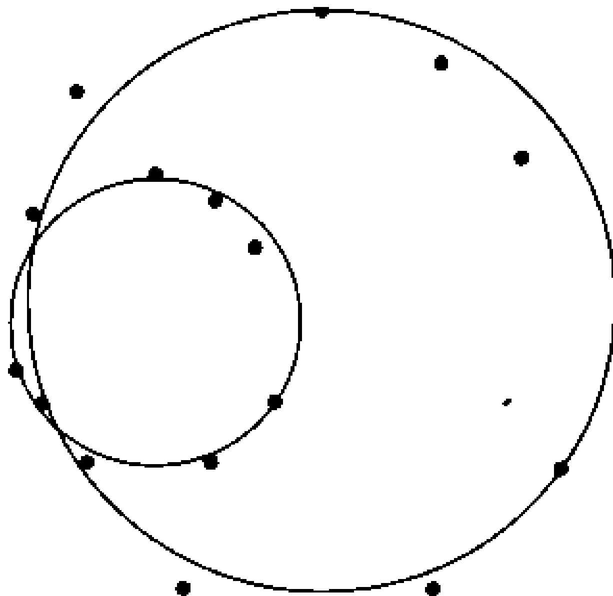


Figura 12.8 Două seturi de puncte aproximate prin cercuri. Punctele de date sunt afișate prin cercuri mici umplute.

Exemplul 12.2: Ni se oferă patru puncte și ni se spune că ar trebui să formeze un pătrat vertical, în timp ce nu formează de fapt unul. Fiecare punct trebuie mutat într-o nouă locație pentru a forma un pătrat, menținând în același timp cantitatea totală de mișcare la minimum. Trebuie să alegeți x,y și *a* pentru a minimiza următoarea expresie:

E = (x,-x) ^{2} + (Ji-^) ^{2} + (x $_2$ -x) ^{2} + O^-ya) ^{2} +

(xj-xa) ^{2} + (j'j-j'-ap + ^-xa)^ (y^^ (12,28)

Luând derivatele parțiale, constatăm că x, y și *a* trebuie să satisfacă următorul sistem:

4x + 2a = 2 x,
i

4^ + 2a = J z

2x + 2y + 4a = y2 + * J + .F3 + *4

Soluția acestui sistem este

a „JP^^r^^^xr^z] (12.29a)

* „ ^(x^x?) + ^(xj+x«) - yto+n) + yC^+n) (12.29b)

y - yfn+y*) + Ytv2+^3) - ^(xj+x«) + v (x >+x 2) (12.29c) o o o
g

Figura 12.9 prezintă exemple de aplicații ale acestor formule. □

Figura **12.9 Două seturi de puncte de date (cercuri umplute) aproximate prin pătrate**

Exemplul 12.3: Dorim să potrivim o elipsă la un set dat de *n* puncte. Fie a și *b* mărimea axelor și <£ unghiul format de una dintre ele cu axa x. Selectăm din nou centrul de greutate (x f ^y f) ca centru al elipsei. Calculăm momentele de inerție față de acel centru:

w„ - su-xj ^{2} , ^»j^-yj ^{2}
rl /-I

şi

^ - 5 la „x^th -^ >
I-1

Se poate arăta că dacă selectăm 0 ca jumătate din unghiul cu tangenta *Mjy/^Myy-M„),* atunci momentele A/^ și *M^,* față de axele elipsei, vor fi maxime într-o direcție și minime în

altă direcție. *(M^* va fi zero.) Ne stabilim ca scop satisfacerea ecuației elipsei în medie, adică

^ / ^ + V~ }) ~° <,2 ' 30)

unde coordonatele *x* și *y* sunt calculate în raport cu un nou sistem de coordonate având centrul în centrul elipsei și axele paralele cu axele elipsei. Ecuația (12.30) este echivalentă cu

4 ML + -«- °

Poate fi mulțumit de

** " VM^/P* B " y/M^/qn .*

unde

p +q - 1 .

Dacă alegem

apoi păstrăm forma distribuției: o elipsă cu aceste axe și densitate uniformă a masei de-a lungul circumferinței sale are momente în raport cu ambele axe care sunt egale (în cadrul unui factor de scară) cu momentele originale. □

12.7 NOTE BIBLIOGRAFICE

Aproximarea prin polinoame este un subiect standard în majoritatea textelor de analiză numerică, cum ar fi (I0.DA), I10.IK] și [10.R1]. Aproximarea prin spline cu noduri fixe poate fi găsită în [1I.DB3]. [ll.LCS], etc. Deși spline cu noduri variabile sunt una dintre cele mai bune alegeri posibile pentru aproximarea punctelor de date cu o curbă netedă,

utilizarea lor este dificilă, iar literatura de specialitate este destul de limitată. 112.MCI descrie unele dintre proprietățile lor fundamentale. O soluție suboptimală este prezentată în (I2.DB). Un anumit algoritm este descris în (I2.R1). Lucrarea respectivă este interesantă nu numai pentru algoritmul prezentat, ci și pentru statisticile de calcul referitoare la gradul polinomului, numărul de noduri și eroarea de aproximare. Aproximațiile poligonale în care locația și numărul de colțuri sunt variabile sunt mai simple și sunt discutate în (3.PA], [12.PA] și [12.PH). Aceste referințe conțin descrieri extinse ale algoritmilor de împărțire și îmbinare. [I2.PH] introduce această abordare pentru potrivirea curbei, în timp ce [I2.PA] utilizează split-and-merge pentru

inițializarea metodei lui Newton, (3.PA) conține descrierea multor dintre metodele euristice, inclusiv tehnicile de scanare, care au fost propuse de diverși autori. Metodele propuse de Tomek (12 TO] și poligoanele de perimetru minim 112.SK] sunt mai apropiate de ideea algoritmului 12.2.

12.8 LITERATURA RELEVANTA

[I2.DB] de Boor. C. „Aproximare bună prin spline cu noduri variabile". *ISNM.* **2! (1973) pp. 57-72.**

[12. MC) McClure. DE „Aproximarea funcției segmentate neliniare și analiza modelelor de linii". *Trimestrial de Matematică Aplicată.* **33 (1975) pp. 1-37.**

(12.PA) Pavlidis, T. „Aproximații poligonale prin metoda lui Newton". *IEEE Trans. Calculatoare.* **C-26 (1977), p. 800-807.**

(12.PH] Pavlidis. T. şi Horowitz. SL „Segmentation of Plane Curves", *IEEE Trans. Computers.* **C-23 (1974). pp. 860-870.**

[12.RI) Rice, J, R.. „On Adaptive Piecewise Polynomial Aproximation," *în* **II I.LSI. p. 359-386.**

[12.SK] Sklansky. J.; Chazin, RL; și Hansen. B J. „Poligoane perimetrale minime de siluete digitalizate". *IEEE Trans. Calculatoare .* **C-21 (1972), p. 260-268.**

(12.TO) Tomek, I. „Piecewise Linear Aproximations", *IEEE Trans. calculatoare .* **C-23 (1974). pp 445-448.**

12.9 PROBLEME

12.1. Verificați valorile elementelor și determinantul matricei Gram prezentate în Secțiunea 12.3. De exemplu, arată asta

tA.iUW.tU) -

$$\int_{iL}^{iL+L}\left[\frac{x-iL}{L}\right]^2dx+\int_{iL+L}^{iL+2L}\left[\frac{iL+2L-x}{L}\right]^2dx=\frac{2L}{3}.$$

etc.

12.2. Scrieți un program care implementează algoritmul 12.2.

12.3. Demonstrați afirmația din Secțiunea 12.5.1 despre relația dintre eroarea maximă

obținută de un interpolant și cea a aproximării optime. (Dacă aveți probleme cu dovada, verificați referința (11.DB2I.)

12.4. Modificați Editorul de puncte al Problemei 10.6 pentru a permite adresarea arcului și utilizați-l pentru a se potrivi poligoanelor prin încorporarea rezultatelor Problemei 12.2.

12.5. Demonstrați că potrivirea suboptimă a cercului din Exemplul 12.1 necesită ca r "

W\Zr^ f)2 + ^i^ „^^

12.6. Deduceți ecuațiile (12.29) direct prin minimizarea £ în ecuația (12.28). Există o presupunere de bază despre ordinea punctelor de date? Modificați algoritmul astfel încât să nu fie sensibil la ordine. De asemenea, extindeți-l pentru aproximări prin dreptunghiuri.

12.7. Editorul de puncte poate fi extins pentru a emula un desenator prin încorporarea diferitelor rutine de aproximare, astfel încât schițele neglijente să fie transformate în desene îngrijite. Faceți acest lucru încorporând algoritmul 12.2 în editor. (Această problemă este o extensie a Problemei 12.4.)

12.8. Implementați formulele din Exemplul 12.3 și folosiți-le ca parte a unui editor pentru a desena elipse care se potrivesc cu schițe brute ale curbelor care trebuiau să fie elipse.

Capitolul 13

MONTAREA DE SURFACE ȘI AFIȘAREA SUPRAFĂȚEI

13.1 INTRODUCERE

Afișarea suprafeței unui obiect tridimensional este o problemă importantă în aplicațiile sistemelor grafice în care scopul este de a oferi utilizatorului vederi diferite ale unui grup de obiecte solide. Potrivirea la suprafață este, de asemenea, necesară atunci când obiectele sunt date inițial ca seturi de puncte. Montarea suprafeței și afișarea sunt de asemenea de interes în alte domenii. În cartografie sau geografie computerizată se ocupă de modele de teren care sunt exprimate ca suprafețe matematice. În procesarea imaginii și recunoașterea modelelor, suprafețele intră în analiza imaginii în cel puțin două moduri. În primul rând, se poate gândi la o imagine ca la o suprafață utilizând valorile de luminozitate pentru a treia coordonată. În al doilea rând, s-ar putea dori să ia în considerare suprafețele obiectelor care apar într-o imagine. Apoi problema de segmentare (Capitolul 4) se reduce la identificarea unor grupuri de pixeli care sunt imagini ale unei singure suprafețe.

În toate aceste aplicații, suprafețele implicate sunt prea complexe pentru a fi descrise printr-o singură ecuație pe domeniul lor și, prin urmare, apare nevoia de aproximări pe bucăți. Cele mai simple aproximări sunt liniare pe bucăți și în special poliedre cu fețe triunghiulare. Fie {P| fi o mulțime de puncte situate pe o suprafață 5. Atunci fiecare triplet de puncte definește un plan, iar o alegere adecvată a tripleților constituie o aproximare poliedrică a lui S. Din păcate, în multe cazuri,

numărul de triunghiuri necesare pentru o aproximare rezonabilă este prea mare și trebuie luate în considerare suprafețele de ordin superior. Acest lucru duce la problema cum pot fi reprezentate astfel de suprafețe pe bucăți.

O soluție este de a defini un set de curbe pe o suprafață și apoi de a defini *pete de suprafață de interpolare* peste aceste curbe. Discutăm astfel de metode în secțiunile 13.4, 13.5 și 13.6. O altă soluție este utilizarea *punctelor de ghidare* sau *a planurilor de ghidare* în modul făcut în plan pentru curbele Bez ier și B-splines. Aceste tehnici sunt discutate în Secțiunile 13.7 și 13.8. Secțiunile 13.2 și 13.3 prezintă câteva formule de bază pentru studiul tuturor acestor suprafețe. Secțiunea 13.9 discută problema afișajului, cu accent pe umbrire.

O considerație importantă în selectarea unei forme matematice pentru reprezentarea suprafețelor este ușurința cu care se pot rezolva unele probleme importante în grafică, cum ar fi găsirea intersecției a două suprafețe . În mod clar, avioanele sunt cele mai bune în acest scop, dar pentru un număr mare de patch-uri, cantitatea totală de calcul poate fi substanțială. Suprafețele B-spline au proprietatea cocăi convexe (Propoziția 11.2) care permite definirea planurilor care delimitează locația suprafeței. Numărul de astfel de planuri poate fi semnificativ mai mic decât numărul necesar pentru o aproximare poliedrică și, prin urmare, este mai ușor de rezolvat pentru intersecții și probleme similare în ceea ce privește astfel de planuri. Cantitatea suplimentară de muncă pentru a găsi intersecția exactă este adesea foarte mică. Într-adevăr, cele mai multe dintre corpurile convexe nu vor intersecta toate și, prin urmare, nu este nevoie să verificați dacă suprafețele respective se intersectează. În cazul hărților lerrain pentru utilizare în cartografie computerizată . avioanele par a fi o alegere rezonabilă, deoarece cerințele de netezime nu sunt severe.

13.2 CATEVA PROPRIETATI SIMPLE ALE SUPRAFETELOR

În general, o suprafață este descrisă de una dintre următoarele trei forme de ecuații:

Explicit: z-/(x^) (13.1a)

Algebric: F(x^,z) - 0 (13.1b)

Parametric: *x ^ X(u ,v), y ^ Y(u ,v). z. ^ Z(u ,v)* (13.1c)

Forma paramelică este cea mai convenabilă pentru aplicațiile grafice și va fi cea mai des folosită în acest text. În multe probleme este util să cunoaștem egalitatea normalei la suprafață într-un punct. Acest lucru poate fi găsit cu ușurință din forma algebrică. Deoarece *F(x^a)* este o constantă, diferența sa totală va fi zero, astfel încât

àx dy $\frac{dF}{dz}$ **0** (13-2)

Cantitatile *dx. dy. d:* denotă variații de-a lungul suprafeței, astfel încât acestea pot fi considerate ca definind un vector tangent la suprafață. Ecuația (13.2) poate fi apoi interpretată ca produsul scalar al unui vector ale cărui componente sunt derivate parțiale cu un vector tangent la suprafață . Deoarece produsul este zero, normala la suprafață este paralelă cu primul vector. (Comparați și discuția din Secțiunea 10.7.1.)

Exemplul 13.1: Ecuația unui plan este dată ca

$$bou + fry + cz " 1 .$$

Normalele acestuia sunt paralele cu *(a,b,c).* □

Exemplul 13.2: Ecuația unei sfere centrate la origine este $_x 2 _+ / + Pr\backslash$

Normala în fiecare punct (xj^z) este paralelă cu vectorul *{x^yz).* □

Pentru a studia reprezentarea parametrică introducem următoarea noțiune

OU

Notație similară este utilizată pentru celelalte variabile. Vectorii $(X_{ut} Y_u .Z_y)$ și (^.K.ZJ sunt tangenți la suprafață, astfel încât dacă $(F_x .F_y .F_t)$ este vectorul normal trebuie să avem următoarele două equa
cțiuni:

$$F_x X_y + F,Y_y + F_2 Z_y -0 \tag{13.4a}$$

$$F_x X, +F_y Y, +F,Z_V =0 \tag{13.4b}$$

Se poate arăta prin substituție că cele două ecuații de mai sus au următoarea soluție

$$F_x - Y^-Z'Y, \tag{13.5a}$$

$$F, --X^Z, + Z_U X„ \tag{13.5b}$$

$$F,\sim X_U Y,-Y_U X_V \tag{13,5c}$$

Deoarece sistemul de ecuații (13.4) are mai multe necunoscute decât ecuații, soluția nu este unică și trebuie avut grijă să eliminați toți factorii comuni din partea dreaptă a ecuațiilor (13.5). Acest pas este necesar deoarece ecuațiile (13.4) definesc direcția gradientului, dar nu și dimensiunea acestuia. Această observație este utilă pentru scrierea unei ecuații care conectează gradientul formelor parametrice și explicite. Într-adevăr, forma explicită este un caz special al algebricii, după cum se poate vedea scriindu-l ca

$$2-f(x.y) = O \quad (13.6)$$

Then we find that

E f. I (13.7)

A treia ecuație impune o scădere: constraint so that we end up with the fol-

$Z_„Y_V \sim Z,Y_U$

$X_U Y_V \sim X,Y_U$ (13.8a)

$X_u Z_y \sim X,Z_U$

(13.8b)

Exemplul 13.3: O definiție parametrică a unei sfere centrate la origine este

x — cosucosv *y* " sinucosv z " sinv

Ecuațiile (13.5) dau următoarele valori pentru normal după eliminarea factorilor comuni

F_x - cosucosv *F?* = sinucosv F, = sinv

Sunt aceste valori în acord cu ceea ce se așteaptă de la Exemplul 13.2? Găsim și asta

tanv ^{Jy} tanv

□

13.3 PUNCTE SINGULARE ALE unei suprafețe

Dacă o suprafață este dată într-o formă explicită, punctele în care derivatele parțiale ale lui z față de x și spre *y* dispar sunt numite puncte *singulare* sau *staționare* . Vom studia proprietățile acestor puncte printr-o expansiune Taylor a funcției *f* în vecinătatea lor. Vom folosi o notație similară cu ecuația (13.3) pentru derivatele parțiale. Dacă AWo ^{arc} coordonatele unui punct singular, avem

$/(\hat{}\hat{}) = f(x_0 .y_0) + (x - x_0)/_x + O' -y\hat{}f, +$

y(x - xo) ^{2} /« + (x - x_0)O " ^o)/^ + „"O' - yo)7 $_{>r}$ + ' • ' (13.9)

/„ este derivata a doua parțială în raport cu x, etc. Al un punct singular ambele derivate parțiale primare sunt zero. Prin urmare, forma lui

funcția este determinată de derivatele a doua. Vom investiga variația funcției de-a lungul unei direcții fixe din ($x_0 j_0$). Se notează raportul O' - >o)/(x - JC_0) cu *h.* Apoi expansiunea Taylor de mai sus poate fi scrisă ca

/(^) „/(* $_0$,y $_0$) + 2(x -XQ) ^{2} ^« + 2A/^ + A ^{2} /^] + '■■ (13.10) presupunând că *x* *XQ. Știm din algebra elementară că un polinom cua dratic are semnul coeficientului termenului său de conducere (adică discriminantul nu are semnul real al rădăcinii sale). cantitatea dintre paranteze (considerată ca polinom în A) va fi întotdeauna aceeași dacă

*f»-f*J„<0* (13.11)

În mod clar, această inegalitate implică, de asemenea, că /„ și *f a* au același semn. Când sunt pozitive, cantitatea din acolade va fi de asemenea pozitivă, iar valoarea funcției departe de singularitate va fi mai mare decât valoarea acesteia la punctul singular. Prin urmare, punctul singular este un *minim.* În schimb, dacă aceste derivate sunt negative, punctul singular va fi un *maxim.* Dacă inegalitatea ecuației, atunci acest *punct nu are* loc . este o direcție de-a lungul căreia punctul singular apare ca minim și o altă direcție în care apare ca maxim.

Exemplul 13.4: Se consideră funcția *z ^f(xy)* -x^2 + *laxy* + $/^2$. În mod clar, punctul (0,0) este singular și cantitatea dintre paranteze din ecuația (13.10) este acum

$$2(1 + 2ah + A^2)$$

Dacă valoarea absolută a lui a este mai mică decât 1, atunci punctul singular este un minim. Într-adevăr, pentru valori mici ale *suprafeței* arată ca o cupă cu secțiuni transversale aproape circulare. Dacă a este un număr pozitiv mare, atunci termenul *xy* devine dominant. Dacă x și *y* au același semn, valoarea funcției va fi mai mare decât zero în cea mai mare parte, în timp ce opusul va fi adevărat dacă x și >* au semne diferite. În special , să considerăm dreapta *x — y.* Atunci z — 2(1 + o)x^2 și pentru a pozitivă intersecția *suprafeței* și un plan de-a lungul liniei va arăta un minim. Dacă x = — *y.* atunci avem *z* = 2(1 - *a)*x^2 și intersecția arată un maxim. Figura 13.1 ilustrează forma suprafeței pentru o = 2. Se poate vedea de ce se folosește denumirea de „punct de șa”.

□

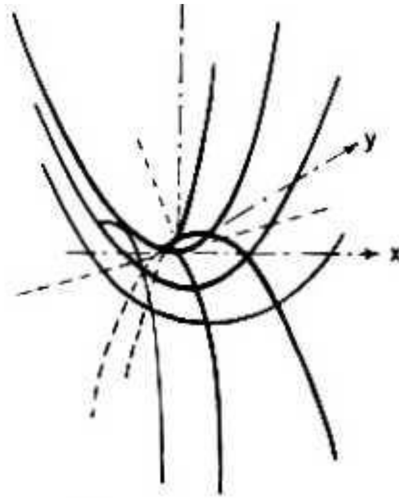


Figure 13.1 A saddle point

13.4 PETICURI DE SURFACE INTERPOLARE LINEARE SI BILINIARE

Cea mai simplă formă de interpolant în trei dimensiuni este triunghiul plan definit de trei puncte *{P f }}.* Fie u și v variabile scalare de la 0 la 1. Atunci suprafața triunghiului cu vârfuri în aceste puncte este dată de

T(uv) - P t u + Pp + Fjd-Mv), (1312)

cu condiția ca u+v<l. În mod clar, 7(1,0) = P,, 7(0.1)-Pj și 7(0,0) " Pj. În plus, 7(u,0) este linia care unește P (și Pj, 7(0.v) este linia care unește P. și *P)t* și 7(u,lw) este linia care unește Pi și P-.

O formă puțin mai complexă implică interpolarea biliniară pe un set de patru puncte (vezi Figura 13.2). Atunci o suprafață biliniară de interpolare S(uv) este definită ca

5(uv) - P|(lu)(lv) + P 2 (lu)v + P 3 u(lv) + P^v . (13.13) unde 0 <uv < I.

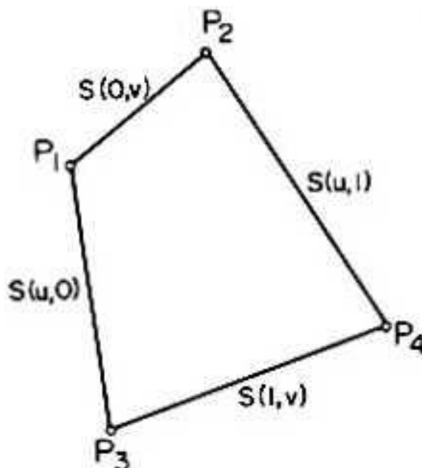


Figura 13.2 Dispunerea celor patru puncte care definesc o suprafață biliniară

Evident 5(0,0) - ?|, S(0,l) - P_2, etc. De asemenea 5(0,v) este segmentul de linie care unește P_t și P_2, 5(1,v) este segmentul de linie care unește P_2 și *P^.* etc. Dacă cele patru puncte sunt coplanare, atunci 5 este patrulaterul plan care le interpolează; în caz contrar 5 este o suprafață de gradul doi. Gradientul suprafeței față de *u* și v poate fi găsit prin diferențierea simplă a ecuației (13.13):

■^-(P.-F,)»+(P J -P,)(lv) (13.14a)

■^-^«-Pj)« +(P J -P|)(1-«) (13.14b)

Rețineți că acestea sunt ecuații vectoriale, astfel încât, dacă *X,* indică coordonata x a punctului P_t, avem

X"-(X4 - X2) v + (A > -X1) (lv).

și în mod similar pentru Y^u și Z". Am folosit un superscript pentru a desemna derivata parțială mai degrabă decât un indice ca în ecuația 13 3 pentru a evita confuzia cu indicele care identifică punctele. Gradientul față de coordonatele x-> poate fi găsit printr-o aplicare a ecuațiilor (138).

Exemplul 13.5: Fie P, - (0,0,0), P_2 = (0.1,0), P_3 = (1,0,0) și *Pi* = (13.0- Atunci coordonatele x, y, z ale fiecărui punct al interpolării biliniare vor fi date de

x(u,v) = u, y(u,v)-v, z(u,v)-uv (13.15a)

sau

z-xy (13.15b)

iar gradientul suprafetei de

|1
|0 (13.16a)

du V,

ds = bi (13.16b)
dv U

Combinația ecuațiilor (13.15a), (13.16a) și (13.16b) are ca rezultat

(13.16c) dx

şi

(13.16d)

care ar fi putut fi găsită și printr-o diferențiere directă a ecuației (13.15b). □

Concluziile exemplului de mai sus au valabilitate și în alte cazuri deoarece se poate alege oricând planul definit de trei dintre puncte ca z = 0 și se plasează unul dintre ele la origine. Apoi, pentru suprafață , z - z 4 uv.

13.5 SUPRAFEȚE LOFTATE

Suprafețele loft sunt produse prin interpolare liniară nu între puncte, ci între curbe. Fie P(0.v) și P(lv) ecuațiile a două curbe spațiale. Apoi o suprafață înălțată este definită ca

S(uv) = (lu)P(0,v) + uP(l,v) . (13.17)

Cu alte cuvinte, suprafața este produsă de un segment de linie care alunecă între două curbe. De exemplu, dacă cele două curbe sunt cercuri în planuri perpendiculare pe linia care le unește centrele, atunci suprafața rezultată va fi un con tăiat de două plane.

13.6 SUPRAFEȚELE COONS

Acestea sunt suprafețe care se interpolează între patru curbe. Ele poartă numele SA Coons, care le-a propus în jurul anului 1960 (vezi Notele bibliografice). Fie *P(u,v)* o funcție a două variabile astfel încât pentru u sau v constantă, P(u,v) se reduce la reprezentarea parametrică a unei curbe spațiale. O zonă de suprafață este construită prin amestecarea curbelor de margine /'(u,0), P(u,l), P(O,v) și P(l,v) în următorul mod:

S(u,v) = P(u,O)(lv) +P(u,l)v + P(O,v)(lu) + P(l,v)w

-P(0,0)(lu)(lv) - P(0,l)(lu)v

-P(l,0)u(lv) - P(ll)uv. (13,18)

Ultimii patru termeni sunt necesari pentru a evita numărarea de două ori a secțiunilor inter-perechi ale celor patru curbe. În acest fel avem:

SOGO) = P(u,0) + P(0,0)(iu) + P(1,0)M

- P(0,0)(lu) - P(l,O)u = P(MO) ,

cu expresii similare pentru S(M,1). etc. Ecuația (13.18) garantează că suprafața rezultată

va fi continuă, dar nu neapărat netedă din cauza posibilelor discontinuități de gradient. Înainte de a discuta cum să asigurăm netezimea, vom introduce o formă mai compactă a ecuației (13.18). Lasă

^'“^O' (13 - 19a)

„W“£M- (13J9b)

şi

6 <“>”M’ (,3,9c)

Apoi ecuația (13.18) devine

S(u,v) = *'00^,0») + P 2 '(u)*(v) - *b'(u)Mb(v)* (13,20)

unde W este o matrice cu *My* = *P(iJ).* Până acum am folosit funcții de amestecare liniare, dar acest lucru poate fi generalizat pentru a permite forme arbitrare. În special, definim

'(“»-[Kij- (132la)

pentru orice funcţii M“) care au proprietatea

M“) $^{=5}$ »< * (13.21b)

pentru orice număr întreg *u.*

Exemplul 13.6: Următoarea este o alegere posibilă pentru funcțiile de combinare :

M") „ ^{cos2} (y“ > (13.22a)

M*) = sin ^{2} (y«) (13.22b)

Dacă curbele de delimitare sunt linii drepte cu colțuri în punctele utilizate în Exemplul 13.5. atunci avem următoarele ecuații pentru coordonatele xy și z:

P(uO)-uP3 , P(M) =MP4 + (1-«)P2.

P(0.v) -vP2 . P(lv) = vP4 + (lv) P3 .

Ecuația suprafeței Coons definită de aceste puncte va fi:

S(uv) = cos ^{2} (-^-v)uP $_{3}$ + sin ^{2} (—v) (MP $_{4}$ + (lu)P $_{2}$ j +

cos ^{2} (yu)vP $_{2}$ + sin ^{2} (ya) JvP $_{4}$ + (lv)P $_{3}$] —

cos ^{2} (-^u)sin ^{2} (^-v)P $_{2}$ - sin ^{2} (-y/)cos ^{2} (^-v)P $_{3}$ -
2 2 2 2

sin ^{2} (yu)sin ^{2} (yv)P $_{4}$.

În special, coordonatele suprafeței vor fi date de

x(u,v)-u, y(uv) = v. (13.23a)

şi

z(uv) = sin ^{2} (yv)u + sin ^{2} (yu)v - sin ^{2} (yu)sin ^{2} (yv) . (13.23b) □

Pentru a avea continuitatea gradientului, putem dori să interpolam nu numai în ceea ce privește curbele P(u,v) ci și în ceea ce privește derivatele lor. Acest lucru poate fi realizat prin adăugarea a două componente suplimentare la fiecare dintre vectorii ecuațiilor (13.19), reprezentând funcțiile derivate și funcțiile de amestecare pentru acele derivate. De asemenea, matricea *M* trebuie extinsă la o dimensiune 4X4. Necesitatea precizării acestor derivate este un dezavantaj al metodei și

complică procesul de proiectare. Formalismul obișnuit în acest caz este de a folosi superscripte pentru a desemna o derivată și în special de a defini pentru orice funcție g(w,v)

£iS3± *du*

(13.24a)

şi

dudv (13.24b)

Apoi definim

/%(*) = | P(0,v) /'(lv) r(o,v r(l,v | (1125)

și la fel pentru /*?(“)• Asemenea

*(*) = | M“) *01(*> ft io(" > ^ll (">. | (13,26)

cu presupunerea că

bji(u) " ^kj'^iu (13,27)

pentru întregul *u,* și *k* — 0 dacă nu se ia nicio derivată și & = 1 dacă există diferențiere. Astfel,

M) = l (13.28a)

*oi(D = *oo(O) = *&>(!) = 0. (13.28b)

Cu alte cuvinte, 6 0((u) este funcția de amestecare pentru curbe și t 1((u) funcția de amestecare pentru pante. 5 t/ este unul în următoarele două cazuri: *k = j = 0* corespunzător primelor două componente ale vectorului de amestecare, iar *k = j = I* derivatelor ultimelor două componente . Matricea M este acum dată de matricea *M.*

P(0,0) P(0,l) P v (0,0) ^(0,1)

P(l,0) FO.!) P*(l,0) r(ll)

M „ p“(o,o) P“(OD r<oo) r(o,n (l3,29 >

/»“do) p^ii) P^UO) /'““(ii)

Cu aceste notații Ecuația (13.20) rămâne valabilă. Mai mult, se poate verifica că nu numai egalitățile de tip S(u,0) = *P(u,Q)* sunt

valide, dar sunt valabile și egalități de formaS“(u,O) = P“(u,0), etc.

13.7 SUPRAFETE DIRIGATE

Este posibil să se specifice o suprafață în termeni de puncte de ghidare printr-o simplă generalizare a polinoamelor Bezier sau a B-spline ale formei discutate în Secțiunea 11.6.

13.7.1 Suprafețele Bezier

Dacă *Q(t)* și *R(t)* sunt două polinoame Bezier, atunci o suprafață S(uv) poate fi definită ca produsul lor tensor 0(u)P(v). Acest produs este evaluat separat pentru fiecare coordonată a punctelor de ghidare. Această procedură nu are efect asupra formei în care apar variabilele *u* și v, dar afectează forma coeficienților. Dacă setăm P* = $Q_t^k R_j$, unde *h* reprezintă x, *y* sau z, atunci ajungem la următoarea definiție:

5(u,v) = 22 *C?u'(tu}"- ¹ Cp'(]-v) ⁿ -JP t/ .* **(13.30)** *iQ J-0*

Este ușor să verifici asta

S<UO) - *^cru'Ou)"-'^ ,* **(13.31)**
1-0

și pentru a obține expresii similare pentru S(u,l), 5(0,v) și S(l,v). Astfel 5(u,v) este mărginit de patru polinoame Bezier și cele patru puncte de colț sunt Poo, *P^* ^o. și P^. Se poate arăta că suprafața se află în carcasa convexă a punctelor de ghidare observând că suma dublă a coeficienților lui $P_{t)}$ este egală cu I.

Este, de asemenea, posibil să se descrie suprafața în termenii părților sale într-o manieră similară cu cea folosită în Secțiunea 10.5 pentru polinoamele Bezier (Teorema 10.1.) O astfel de descriere este utilă în aplicațiile grafice în care s-ar putea dori să subdivizați un patch în părți mai mici pentru a rezolva problemele de vizibilitate (vezi Capitolul 17).

13.7.2 B* Suprafețe Spline

O suprafață B-spline este definită formal ca

ff«,»-)- 2 2 'W■')*,>,(>')P„ , **(13.32)**
i—-* *J—m*

pentru $u_0 < u < M^*_{+l}$ și $v_0 < v < v_{/+l}$. Fără pierderea generalității presupunem că u_0 " v_0 - 0. Reamintim că *N* $_{ijn}$ *(u)* este zero, cu excepția cazului în care $u_t < u < U_i + _n + i$ și că suma tuturor B-spline-urilor dintr-un anumit punct este l (Teorema 11.1). Apoi putem găsi că

* 0
P(B,O)- 5 ^^(u) 2 A'^IO)^
i—-m y—-m

Dacă punctele de ghidare cu indici negativi sunt toate aceleași cu $/^*_{/0}$, atunci ecuația de mai sus este simplificată în

P(uM= 5 ^(«^o. (13-33)
fm

1 .e., suprafața este mărginită de o spline pentru v — 0. Aceeași metodă și ipoteze conduc la rezultate identice pentru celelalte valori extreme ale lui *u* și v. Desigur, acest rezultat nu este la fel de important ca în cazul suprafețelor Coons, deoarece ecuația (13.32) nu descrie un singur petic, ci întreaga suprafață de interes. Patch-urile pot fi găsite pentru fiecare set de valori ale lui *u* și v care se află între punctele de întrerupere: Mj < u < M /+ |, Vy < v < Vy+|. Este instructiv să studiem forma ecuației (13.33) pentru unele cazuri speciale.

Luăm în considerare mai întâi interpolarea biliniară când *m* = I și punctele de rupere sunt distribuite uniform, la o unitate de lungime între ele (Z. — 1 în termenii ecuației (11.8)). Atunci avem

P(u,v) - (u-ujfv-v;)^ + (uw^Xvy+iv^ji

+ (uy+i-uMv-VjJP'-ij + (Mf + |-u)(v y+ i"v)A-ij- 1 • (13.34)

Aceasta este o formă biliniară similară cu cea dată de ecuația (13.13). Deoarece u /+ i — u y — 1 și v /+1 — v, = I aflăm că

P(u M v) - (vv y)Au + (v >+ |-v)P / _ u _| , (13.34a)

P(M /+H V) - *(v-Vj)P If* + (v y+ -^/ji , (13.34b)

etc De asemenea

^(^.vp- A-IJ-I . (13.35a)

P^.Vj+d-Pij . (13.35b)

etc. Prin urmare, interpolarea biliniară este un caz special al ecuației (13.32) forma - I.

În continuare studiem suprafețele obținute prin B-spline uniforme de gradul doi (m =2) cu £ -1. De asemenea, alegem u 0 = 0 astfel încât *u t - i.* Am văzut în Secțiunea 11.7 (Exemplul 11.4) că astfel de spline sunt potrivite în special pentru descrierea formei. Analiza utilizată pentru a deduce ecuațiile (11.30a) și (11.32) poate fi abrogată în cazul de față pentru a obține expresii similare. În special, constatăm că

Pku^vj) = y^-ui + AU-2 + A-JJ-I + ^-2^-2] .(13,36)

adică suprafața trece prin centrul patrulaterului format din patru dintre punctele de ghidare. Asemenea

-^P *\u,Vj* - y(P,-ij-i + P<-ij-2> " y(^-2j-i + *Pi-ij-2*> - (13-37) Ultimele două ecuații înseamnă că, dacă se formează un poliedru ale cărui laturi sunt patrulatere și ale cărui vârfuri pot fi mapate într-o rețea pătrată, atunci suprafața liniilor de ghidare are drept ghidaj suprafața punctelor prin punctele de trecere. centrul fiecărei fețe și este tangentă la aceasta în acel punct.

13.8 ALEGEREA UNUI DESPARTIZARE DE SUPRAFAȚA

Dacă o suprafață este dată de un set de puncte, trebuie să alegeți o partiție

adecvată sau să definiți setul de puncte de ghidare înainte de a aplica oricare dintre metodele din cele patru secțiuni anterioare. Aceasta este generalizarea problemei locației nodurilor din Secțiunea 12.4. Având în vedere dificultatea acestuia din urmă, nu este surprinzător că nu există soluții automate disponibile pentru problema actuală. În multe aplicații grafice, curbele sunt desenate pe un model solid, iar acestea servesc ca intrare în timpul digitizării . În procesarea imaginilor, găsirea partiției este echivalentă cu rezolvarea problemei de segmentare (Capitolul 4). Deoarece constrângerile de continuitate nu sunt importante în acea aplicație, problema este oarecum simplificată.

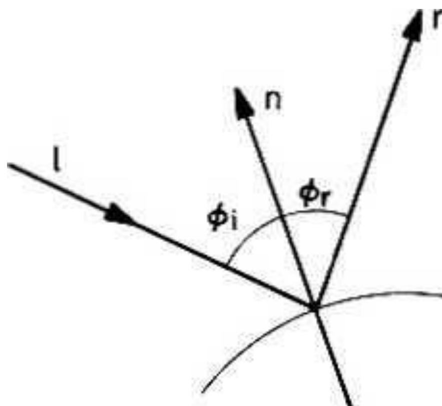
13.9 AFISARE SUPRAFEȚE ȘI Umbrire

Problema noastră majoră în afișarea curbelor a fost conversia formei matematice într-un set de coordonate pixeli. Aceste informații sunt al) de care aveți nevoie pentru a afișa o curbă plană. A avea coordonatele (x 4 ^.z ( ) ale punctelor unei suprafețe este departe de a fi suficient pentru o afișare. Sunt posibile două abordări de bază: codificarea informațiilor de înălțime (coordonata z) în culoare sau umbrire. Codificarea culorilor duce la afișaje de contur care pot fi utile pentru unele aplicații, dar nu sunt nici atractive din punct de vedere estetic, nici informative. Umbrirea poate crea un plan de calcul mai bun, dar dacă toate oferă planuri de calcul mai bune. (impresia de umbrire prin acordarea unei culori uniforme sau a unui nivel de gri proiecției fiecărui plan vizibil. Uneori, aceasta este suficientă pentru a crea iluzia de adâncime. Unele dintre problemele în afișarea suprafețelor neplane apar deoarece trebuie să ne ocupăm de proiecțiile lor pe un plan și, prin urmare, trebuie să ne ocupăm de problema vizibilității: unele dintre punctele de suprafață le pot ascunde pe altele. În special, proiectarea suprafeței nu este întotdeauna proiectarea unei suprafețe. Un tratament complet al acestor probleme este în afara noastră

domeniul de aplicare, așa că ne vom limita discuția la principiile de bază.

În grafica raster, contururile pot fi afișate într-un mod simplu prin alocarea unei corespondențe între *2* și nivelul de culoare sau gri. Dacă punctele au fost selectate cu o densitate suficient de mare, contururile constantei z *vor* apărea ca curbe de luminozitate constantă. Problema este mult mai dificilă în grafica vectorială, deoarece trebuie să urmărim astfel de curbe pe suprafață.

Umbrirea presupune existența unei surse de lumină precum și cunoașterea proprietăților reflectorizante ale suprafeței. Dacă sursa de lumină este foarte departe de obiect, atunci se poate presupune că razele de lumină sunt toate paralele și aspectul unei suprafețe va depinde de proprietățile ei proprii și de orientarea față de sursa de lumină. Cantitatea de lumină incidentă pe o suprafață este proporțională cu produsul scalar al normalei » la suprafață și cu un vector / paralel cu direcția sursei de lumină, așa cum se arată în Figura 13.3. (În Secțiunea 13.2 am văzut cum să găsim normalele suprafețelor definite matematic.)



Figare 13.3 Reflectarea luminii de pe suprafețe, *n* denotă normalul față de suprafață, *l* direcția sursei de lumină și *r* direcția luminii reflectate. ÿ< este unghiul de incidență și *4»* *r* unghiul de reflexie.

Dacă o suprafață este o oglindă perfectă, atunci luminozitatea ei în fiecare punct poate fi calculată din legea fundamentală a reflexiei: unghiul de reflexie este egal cu unghiul de incidență. Lumina va fi văzută de observatori numai dacă privesc din direcția potrivită. Fie e un vector paralel cu acea direcție și să presupunem că acest vector, precum și /. a fost normalizat la lungimea unității. Apoi bisectria unghiului lor va fi paralelă cu suma lor /+e, astfel încât cantitatea de lumină văzută variază cu produsul scalar al acelui vector și normala la suprafață, n. The

suprafața va apărea cea mai strălucitoare când unghiul este zero și întunecată când unghiul este egal cu 90'. O astfel de reflexie se numește *speculară* și contribuie doar parțial la luminozitatea totală a suprafeței. Reflexia *difuză* contribuie la luminozitate suplimentară. Aceasta presupune că lumina este reflectată în mod egal în toate direcțiile, iar diferențele de luminozitate se datorează doar diferențelor în cantitatea de lumină incidentă. Prin urmare, iluminarea datorată reflexiei difuze este proporțională cu produsul scalar al *lui l* și n. În mod clar, dacă produsul scalar este negativ, nu există deloc lumină reflectată, astfel încât în calculul nostru ar trebui să luăm în considerare produsul scalar numai dacă este pozitiv. Dacă I_t denotă reflexia speculară și ! $_d$ reflexia difuză pe care o avem

I, - max(0,eu] (13.38a)

I< - max(0,/*] (13,38b)

Pentru a calcula iluminarea totală trebuie să formați o sumă ponderată a acestor termeni. Mai mult, există dovezi empirice că ridicarea lui *I,* la o anumită putere *k,* produce evidențieri mai realiste decât în alt mod (vezi Notele bibliografice). Astfel avem

I-mh+glf (13,39)

unde m este un coeficient care descrie cât de mată este suprafața și *g* un coeficient care descrie cât de lucioasă este. În mod clar, ceea ce este important este dimensiunea

relativă a lui *m* și *g.* Unii autori le selectează sub constrângerea m + g = I. Valorile lui *k* în intervalul 5 până la 60 au fost folosite pentru a produce câteva imagini impresionante raportate în literatură (13.BN).

Aplicarea practică a acestor formule necesită calculul normalului în fiecare punct al suprafeței. Au fost raportate diferite aproximări care calculează valoarea doar în câteva puncte și se intercalează între celelalte (vezi Notele Bibliografice).

Exemplul 13.7: Vom folosi expresia pentru normala la o sferă, găsită în exemplele 13.2 și 13.3, pentru a construi o afișare umbrită a unei sfere. Presupunem că suprafața este iluminată de sus, astfel încât / - (0,0,1), că un observator este de-a lungul axei x și că afișajul este pe planul *yz* . Atunci e - (1,0,0) și suma normalizată a acestor doi vectori va fi (l/VW/VS). Deoarece M -(x^. z) aflăm că

$$\left[0, \frac{x+z}{\sqrt{2}}\right] \qquad /\text{^ - max } [0,z\ ]$$

^"«Ofilie definind ecuația pentru o sferă, avem x - ±Vl-"p + z l , astfel încât ecuația (13.39) devine

/-mmax[O,z]+g[max[O^ ±\/T : ?+7]] (13,40)

Această ecuație poate fi evaluată cu ușurință pentru diferitele valori ale lui *y* și *z* care satisfac j ^{2} + z ^{2} < 1 (vezi problema 13.3). Figura 13.4 (Plansa 27) prezintă două vederi ale sferei, ambele pentru Ar — 1. □

13.10 NOTE BIBLIOGRAFICE

Afișarea și descrierea suprafeței este un subiect care a primit atenție în literatura de specialitate în cea mai mare parte din 1970. Popularitatea sa a coincis cu disponibilitatea graficelor raster la costuri reduse. Lucrările anterioare s-au concentrat în primul rând pe descrierile matematice.! Pentru un studiu timpuriu al suprafețelor Coons vezi [13.FO]. [13.LR] este un studiu recent care pune accent pe suprafețele Bezier și B-spline. [ll.BR], [13.LA] și [13.SSS] subliniază utilizarea suprafețelor în aplicațiile grafice. Datorită naturii foarte locale a reprezentării, suprafețele B-spline par cele mai promițătoare pentru aplicațiile în care suprafața poate fi proiectată interactiv. [13.BL] este o referință foarte bună pentru afișarea pe suprafață și discuția din Secțiunea 13.7 s-a bazat pe aceasta. [13.BN] prezintă o mică parte din materialul [13.BN], dar conține numeroase exemple și este ușor de găsit într-o bibliotecă. Modelul ecuațiilor (13.38) și (13.39) a fost propus în [13.PH]. Vezi [13.WH] pentru un model mai general care include și lumina transmisă și [13.CT] pentru un model care ține cont de dependența reflectanței de lungimea de undă a luminii. [13.LA] conține un tratament cuprinzător al problemei de afișare a suprafeței, incluzând strategii de evaluare a normalului, a vizibilității și a umbririi. [13.HO] este o trecere în revistă a unui subiect înrudit, umbrirea hărților de teren. Ar trebui să subliniem că regulile de umbrire folosite de artiști sunt diferite de regulile matematice descrise în Secțiunea 13.9. Ei presupun adesea că iluminarea difuză se datorează reflexiilor secundare ale sursei de lumină, iar zona cea mai întunecată este o zonă între evidențierea reflexiei speculare și maximul reflexiei difuze (vezi [13.GU], de

exemplu).

13.11 LITERATURA RELEVANTA

[I3.BL] Blinn, JF „Afișarea de computer a suprafețelor curbe”. *Ph.D. disertație* , Departamentul de Informatică, Universitatea din Utah, 1978.

[13.BN] Blinn, JF și Newell, ME „Textură și reflexie în imagini generate de computer”. *CACM,* **19** (1976), p. 542-547.

t Desigur, artiștii s-au confruntat cu problemele descrierii suprafeței și umbririi timp de secole, dar metodele lor nu sunt supuse implementării imediate de către computere.

[I3.CT) Bucăt. RL și Torrance. KE *'A Reflectance Model for Computer Graphics." *SIGGRAPH81.* **Dallas. Texas. (august 1981), pp. 307-316.**

II3.FO] Forrest. AR „On Coons and Other Methods for the Representation of Curved Surfaces”, *CGIP.* **1 (1972), p. 341-359.**

(13 GU) Guptili. AL *Desen cu creion și cerneală.* **editie revizuita. New York: Van Nostrand, 1968.**

(13.HO) Corn. B KP „Umbrirea dealurilor și harta reflectanței”. *IEEE Proceedings.* **69 (1981). pp. 14-47.**

(13.LA) Lane. JM; Carpenter, LC; Whitted, T.; și Blinn, JF „Scan Line Methods for Displaying Parametrically Defined Surfaces”, *CACM.* **23 (1980), p. 23-34.**

[13.LR] Lane, JM și Riesenfeld, RF „A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces ”. *IEEE Trans, on Pattern Analysis and Machine I niellige nee.* **PAMI-2 (1980), p. 35-46.**

[13.PH) Phong. BT „Iluminare pentru imagini generate de computer”. *CACM.* **18 (1975). p. 311-317.**

[13.SSSI Sutherland, 1. E.; Sproull, RF; și Schumacker, RA „A Char acterization of Ten Hidden-Surface Algorithms”. *Sondaje ACM Computing.* **6 (1974). pp. 1-55.**

[13.WHJ Whitted. T. „Un mod de iluminare îmbunătățit) pentru afișaj umbrit”, *CACM.* **23 (1980), p. 343-349.**

13.12 PROBLEME

13.1. Arătați că expresiile pentru normala la o sferă obținute în exemplele 13.2 și 13.3 sunt echivalente.

13.2. Studiați forma zonei de suprafață din Exemplul 13.5 prin reprezentarea grafică a diferitelor secțiuni transversale ale S(uv), cum ar fi 5(uu), S(ul-u). *S(u.* 1/3), *S(u* $_t$ 1/2), *S(u,* 2/3), etc.

13.3. (a) Scrieți un program care implementează ecuația (13.40) și utilizați-l pentru a produce o afișare a unei sfere umbrite, cum ar fi cea prezentată în Figura 13.4 (Planca 27). Studiați efectul variației parametrilor d *. g* și *k.* (b) Arătați că locurile punctelor care au aceeași iluminare datorită reflexiei speculare sunt elipse. Arătați în mod formal că toate astfel de elipse sunt coninuite în cercul care este

proiecția sferei.

Capitolul 14

MATEMATICA GRAFICII BIDIMENSIONALE

14.1 INTRODUCERE

Un obiectiv major în grafica pe computer este de a crea afișări picturale din descrieri matematice ale imaginilor. Aceasta necesită crearea (de obicei de către computerul gazdă) a unui fișier de afișare care este apoi trimis către dispozitivul grafic. În grafica vectorială, în care dispozitivul se așteaptă la comenzi precum „desenați vector”, conținutul fișierului de afișare este în cea mai mare parte instrucțiuni primitive. În grafica raster, instrucțiuni precum „desenați vector” nu sunt primitive și sunt traduse de dispozitiv în configurații de pixeli din memoria de reîmprospătare. Strict vorbind, majoritatea dispozitivelor grafice raster nu au un fișier de afișare. Cu toate acestea, este util să ne gândim în ceea ce privește un astfel de fișier, indiferent de modul în care dispozitivul de afișare îl gestionează. Crearea unui fișier de afișare este banală pentru afișaje foarte simple, dar devine destul de dificilă pe măsură ce complexitatea afișajului dorit crește.

Exemplul 14.1: Putem folosi următoarea secvență de instrucțiuni din setul din Secțiunea 1.7 (Tabelul 1.2) pentru a desena un pătrat de intensitate *L* pe un ecran de afișare, cu un colț la (x_0 ,yo), latura de dimensiunea r și laturile paralele cu axa verticală și orizontală:

înainte (L)
setp(x Q ,y Q)
vec (XQ+J^O) *vec* (XO+J^O+J) *vec (x 0 ,y 0* +s) *vec(x o y o)*

Acest grup de instrucțiuni poate fi numit ca un ***pătrat de procedură** (x Q ,y Q ,s ,L).* Pătratul poate fi șters printr-un apel la pătrat (x 0 , y o» J 'O)- (A se vedea mai jos pentru o discuție despre posibilele efecte secundare ale ștergerii.) Umplerea unui pătrat cu o orientare arbitrară cu o anumită culoare este o problemă mai dificilă și necesită implementarea unuia dintre algoritmii discutați în capitolul 8.

spate (L)
câștig (xo.yoAo^ .^o^) *șterge*

Acesta ar putea fi numit ca un program *fsquare {x Q y Q,s ,L).* □

Acest exemplu ilustrează, de asemenea, o problemă comună cu afișajele grafice. Setul de instrucțiuni nu specifică dacă colțul dat (x 0 ,y 0) va fi în stânga sus, stânga jos, dreapta sus sau dreapta jos. Răspunsul depinde de dispozitiv! Unii dintre ei presupun că originea (0,0) este în colțul din stânga jos, în timp ce alții presupun că este în colțul din dreapta sus. În unele cazuri este posibilă selectarea originii printr-o comandă grafică, în timp ce în altele utilizatorul trebuie să creeze fișierele ținând cont de originea utilizată de dispozitiv.

Exemplul 14.2: Următorul program simplu poate crea iluzia unui pătrat care se îndepărtează de observator.

Algoritmul 14.1 Pătrat în mișcare

1. În timp ce *s* este pozitiv, faceți:
   ÎNCEPE.
2. ***pătrat** (x,y ,s,L)*
3. somn (T)
4. ***pătrat** (x,y,s,0)*
   Descreșteți 5 cu 2 și £ cu 1 și creșteți x și v cu 1. Sfârșit.
6. Sfârșitul algoritmului.

Pasul 2 determină afișarea unui pătrat de dimensiunea *s* și culoarea Z., în timp ce pasul 4 afișează un pătrat de dimensiune similară și culoarea 0, ștergându-l pe cel tocmai afișat.! Instrucțiunea de *somn (T)* cere programului să suspende execuția pentru *T* unități de timp, astfel încât pătratul să poată fi văzut înainte de a fi șters. În multe sisteme de partajare a timpului, timpul de așteptare pentru service plus timpul de execuție a instrucțiunilor sunt suficient de lungi, astfel încât să nu fie necesară nicio întârziere suplimentară. S-ar putea folosi */square* în loc de *pătrat* pentru a produce același efect cu un pătrat umplut. Mișcări mai complicate pot fi programate prin schimbarea pasului 5. Totuși, dacă se dorește introducerea rotației, rutina de bază pentru afișarea pătratului trebuie modificată, deoarece laturile nu vor rămâne paralele cu axele de coordonate. O altă problemă potențială este că pătratul se poate deplasa în afara limitelor afișajului. Trebuie să verificați acest lucru și să modificați comenzile astfel încât să fie afișată doar partea din limitele afișajului. Astfel pătratul trebuie tăiat. (Nu se poate depinde de dispozitiv pentru a face decuparea. Consultați Secțiunea 15.1 pentru mai multe despre acest punct.) □

Exemplul 14.3: Dorim să afișăm un cub cu un colț la (XQO'O.^O) și laturi de dimensiunea 5. Va fi necesar să găsim o proiecție a solidului pe plan. Dacă urmează să fie creat un afișaj în mișcare, atunci s-ar putea să trebuiască găsite multe astfel de proiecții. Dacă se dorește un desen în linie, atunci un program simplu, similar cu ***pătratul,*** poate fi utilizat pentru afișare. Pentru o afișare plină, trebuie mai întâi să decidem care trei dintre cele șase fețe ale arcului cubului sunt vizibile. Dacă este implicat mai mult de un cub, atunci s-ar putea să fie nevoie să verificăm dacă unul dintre ele ascunde altul, așa că trebuie rezolvată o problemă de vizibilitate și mai complexă. □

Aceste exemple ilustrează unele dintre problemele majore care trebuie rezolvate pentru a crea un afișaj. Rotațiile și proiecțiile necesită *transformarea coordonatelor,* ceea ce, la rândul său, necesită verificarea dacă obiectele sunt încă în câmpul vizual și dacă se întunecă unele pe altele. Astfel de probleme pot fi numite în mod colectiv *probleme de vizibilitate.* În acest capitol și în următorul capitol vom discuta mai ales despre afișarea obiectelor bidimensionale, în timp ce grafica tridimensională va fi subiectul capitolelor 16 și 17. Coordonate bidimensionale

+ Implementarea instrucțiunii de ștergere este netrivială. (Reamintim discuția din Secțiunile 1.5 și 1.6.) Într-un sistem de grafică vectorială, ar fi mai eficient să se manipuleze direct lista de afișare. Desigur, acest lucru ar face ca problema să nu fie portabilă între dispozitivele grafice vectoriale și raster. În grafica raster, putem fie să folosim o instrucțiune *de activare* pentru a evita ștergerea fundalului și a altor obiecte, fie pentru a defini o fereastră în jurul obiectului, a șterge întreaga fereastră și apoi a restabili scena. În mod clar, este necesară o cantitate considerabilă de contabilitate pentru crearea și manipularea afișajelor complexe.

transformările vor fi discutate în Secțiunea 14.2. Secțiunile 14.3 și 14.4 prezintă anumite rezultate matematice care sunt utile pentru o serie de probleme de grafică. Procesul de a afla dacă o linie (sau poligon) este intersectată de un alt poligon se numește de obicei *tăiere* și va fi subiectul capitolului 15.

Din punct de vedere matematic, algoritmii din acest capitol sunt foarte simpli și se bazează doar pe geometrie analitică elementară. Cu toate acestea, ele oferă bazele algoritmilor mai complecși care creează afișaje interesante.

14.2 TRANSFORMĂRI BIDIMENSIONALE

O problemă foarte comună în grafică este găsirea locației unui pixel (xy) după ce acesta a fost rotit cu un unghi 0 în jurul unui punct (*0^0) - Fie *r* lungimea vectorului care se unește (x,y) cu (x_0.y_0)- Aceasta nu se schimbă în timpul rotației și, prin urmare, avem următoarele seturi de ecuații (vezi figura 14).

x—x_0-rcos0 yy $_0$-rsin4> (14.la)

X—x_0= rcos(^+0) K-yo — rsin(0+0) (14.1b)

Extinderea funcțiilor cosinus și sinus în ecuația (14.1b) și apoi înlocuirea din ecuația (14.1a). constatăm că noile coordonate (X,K) sunt date în termenii celor vechi prin următoarele ecuații:

AT - x_0+ (xx $_o$)cos0 - O'-y $_o$)sin0 (14.2a)

T "^0 +U”^o)»in0 + 0'->o)costf (14.2b)

Este posibil să scrieți aceste ecuații într-o formă mai elegantă prin definirea *matricei de rotație* în jurul originii *R(d)* ca

*(*) - cos0 —sin# / sin0 COS0 (14.3a)

and the vectors

$$X = \begin{bmatrix} X \\ Y \end{bmatrix} \quad x = \begin{bmatrix} x \\ y \end{bmatrix} \quad x_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \qquad (14.3b)$$

Atunci avem

* "X_O +K(0)(x-x_o) . (14.4)

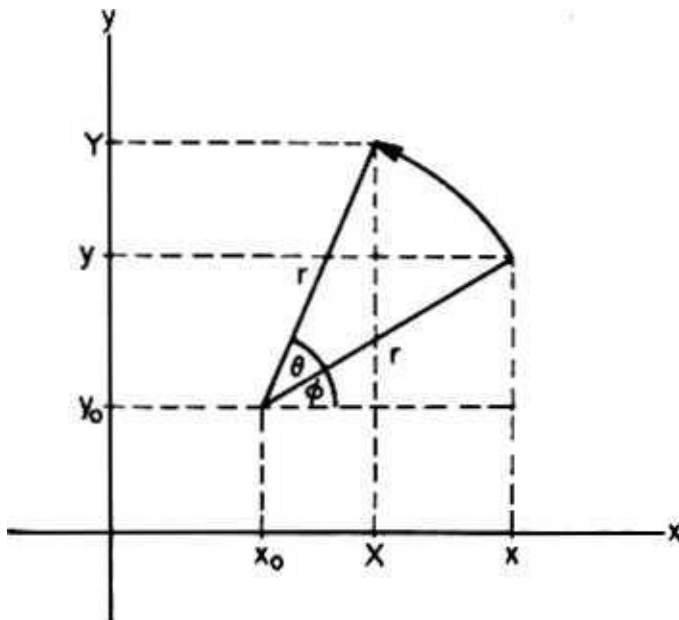


Figure 14.1 **Symbols used for ihe expression of the new coordinates** *(X.Y)* **of a point in terms of the old ones (x^) following a rotation by an angle** *6* **around a point** (XQ^O)

Translația unui punct poate fi exprimată cu ușurință prin adăugarea la coordonatele respective a mărimii mișcării. *Scalare* se realizează prin înmulțirea coordonatelor cu un factor de scalare și poate fi exprimată ca operație matriceală prin definirea matricei de scalare

$$\begin{matrix} S_x & 0 \\ 0 & s, \end{matrix} \qquad (14.5)$$

Există un anumit avantaj de uniformitate dacă exprimăm traducerile prin matrice. Acest lucru se poate face prin adăugarea unei a treia componente egale cu I la definițiile vectoriale ale ecuației (14.3b). Matricele de rutare și scalare pot fi modificate adăugând un al treilea rând și o a treia coloană ale căror primele două elemente sunt 0 și al treilea I, în timp ce o matrice de translație poate fi acum definită ca

$$HAx.Ay) - \begin{matrix} & 0 & Ax \\ 0 & 1 & \hat{} y\ i \\ 0 & 0 & \end{matrix} \qquad (14.6)$$

Acum vom combina aceste matrici într-o singură matrice de transformare. Într-adevăr, unei secvențe de transformări îi corespunde a

înmulțirea matricelor respective și se poate verifica că produsul matricei T(Ax.Ay) cu matricea extinsă /?(?) este egal

ca?	-păcat?	Top
păcat ?	ca?	^y
0	0	1

(14.7a)

Produsul *R(8) expandat* cu T(Ax.Ay) este egal cu cos? -păcat? Arcos?-Da păcat?

păcat? ca? Axsin?+Aycos? .

0 0 eu (14.7b)

Ecuația (14.7a) oferă matricea pentru o translație după o rută.

Ecuația (14.7b) oferă matricea pentru o rută în urma unei translații. Rutarea în jurul punctului (x_0,y_0) poate fi exprimată și ca o succesiune de translații și rutare în jurul originii:

$$X - 7'(x_0 ^ _0)^(?)T(-x_0 -y_0)x \quad (14.8)$$

astfel încât matricea unei rute după unghi ? în jurul unui punct (x_0^ $_0$) urmat de o translație (Ax.Ay), este dat de

MlxoJ'o,?^^) -

ca?	-păcat?	Ax+x_0(l -cos?)—yosin?	
păcat ?	ca?	△ j'+yod- -păcat?)— xoCos?	
0	0	1	

(14.9a)

şi

X-MUo^oAAx.Ay)* . (14.9b)

Ecuațiile (14.9) descriu transformarea generală asupra poziției unui punct pe plan. Merită să subliniem că inversul matricei Af (r_0 .y_0 ,?,Ax ,Ay) poate fi găsit cu ușurință prin înlocuirea Ax și Ay cu -Ax și -Ay și ? cu -?. și inversarea ordinii operațiilor din ecuația (14.8).

14.3 COORDONATE OMogene

Reprezentarea unui punct din plan de către un vector cu trei componente poate fi folosită pentru a face o scară internă în timpul evaluării pozițiilor. Să presupunem că schimbăm unitatea de măsură pentru coordonatele x și y. Atunci a treia componentă nu va fi 1, ci egală cu un factor de scară *h* . și putem scrie

$$\frac{X}{} \quad 9 \quad (14,10)$$

Coordonatele *(hx,hy,h)* se numesc *coordonatele omogene* ale unui punct. Este evident că nici una dintre matricele de transformare nu trebuie schimbată deoarece noua reprezentare corespunde unei înmulțiri cu un scalar.! Principalul avantaj al acestei formulări devine evident atunci când luăm în considerare ecuațiile de linii drepte. O ecuație comună a unei drepte cu panta M este

$$y - nx + v .\qquad (14.11a)$$

Această formă nu poate fi folosită pentru a descrie linii paralele cu axa lor. Pe de altă parte, forma

$$ax + by + c — 0\qquad (14.11b)$$

permite descrierea unor astfel de linii prin setarea 5—0. Înmulțirea ambelor părți ale ecuației (14,1 lb) cu A și stabilirea

t — xh ti – *yh* și f — A

obţinem ecuaţia omogenă pentru o dreaptă

□ f + dj+cf —0. (14.12a)

Dacă definim a ca fiind vectorul coloană cu elementele o, *b* și *c,* putem exprima ecuația dreptei ca produs scalar

o'x=0. (14.12b)

Dacă o mulțime de puncte este transformată printr-o transformare a cărei matrice este *M,* atunci punctele care erau coliniare înainte de transformare trebuie să rămână coliniare. Dacă *A* este vectorul coloană în care *a* fost mapat, trebuie să avem

^'AT-O. (14.12c)

şi prin urmare

X-Noi. (14,13)

De asemenea, observăm că un punct la infinit poate fi exprimat prin stabilirea < egal cu zero. Facilitatea de a manipula punctele la infinit în același mod ca și alte puncte are avantajul practic că punctele cu coordonate în afara intervalului de precizie numerică disponibilă (depășire sau depășire) pot fi manipulate cu ușurință. De exemplu, o linie poate fi definită de două puncte ale căror coordonate sunt în afara intervalului și conțin în același timp și alte puncte, în interval, care trebuie afișate.

t Rețineți că această înmulțire nu introduce o schimbare de scară în afișajul h va fi luată în considerare înainte de crearea afișajului.

Dacă folosim coordonate omogene, coordonatele punctelor definitorii pot fi redimensionate la limite rezonabile. O rutină de tăiere (vezi capitolul 15) poate fi apoi utilizată pentru a găsi partea vizibilă a liniei.

Vom ilustra anumite avantaje matematice ale coordonatelor omogene în timp ce derivăm câteva formule utile.

14.3.1 Ecuația unei linii definite de două puncte

Fie punctele P| și P_2 au coordonatele (x h yi, W|) și respectiv (^2^ >2'^z). Un punct cu coordonatele (x,y,w) va fi coliniar cu ele dacă coordonatele sale sunt dependente liniar de cele ale lui Pi și P_2. Aceasta înseamnă că determinantul unei matrice ale cărei coloane sunt coordonatele celor trei puncte ar trebui să fie zero, adică,

X X| x2 *y* Pi

det 72 w w, w2 - 0 . (14.14)

Aceasta este ecuația dreptei care trece prin cele două puncte P și P_2. Prin extinderea determinantului, putem rescrie ecuația într-o formă mai convențională:

x^^J-wJ-ı) + p($w_1 x_2 - x_1 w_2$) + w($x_L v_2$-y|X_2) -0 (14,14')

(Rețineți că aceasta este echivalentă cu ecuația (10.2).)

14.3.2 Coordonatele unui punct definit ca intersecția a două linii

Punctul de la intersecția a două drepte L, și L_2 cu coeficienți (fl|. 6,. cj și (a_2. b_2, c_2) poate fi găsit într-un mod similar. O a treia dreaptă cu coeficienți *(a, b $_t$ c)* care trece prin acel punct trebuie să fie dependentă liniar de celelalte două. și de aceea o linie determinantă a cărei coloane trebuie să fie zero.

Hc 2

Expansiunea determinantului produce ofé^-c,^) + bk.aj-^cj) + *da^-bia,)* - 0

Deoarece (a. 6. c) sunt coeficienții oricărei drepte care trece prin intersecțiunea lui *Li* și £ 2, termenii din paranteze sunt coordonatele (14.15')

det *b bt b2* -0 . (14.15)

intersecției lor.

14.3.3 Dualitate

Ecuațiile (14.14) și (14.15). sau (14,14’) și (14,15’). demonstrați dualitatea *dintre* puncte și linii de pe plan atunci când sunt utilizate coordonate homogene. Un set de trei numere poate reprezenta fie o linie, fie un punct, astfel încât ecuațiile unei linii definite de două puncte sau cele ale unui punct definit de două linii sunt identice. Principiul dualității este utilizat pe scară largă în geometria proiectivă deoarece, pentru fiecare proprietate a punctelor și a liniilor, se poate obține un alt rezultat prin schimbarea punctului și a dreptei. De exemplu:

Teorema 14.1: Dacă dreptele care unesc vârfurile (punctele) a două triunghiuri trec

printr-un punct comun, atunci punctele în care laturile (liniile) respective se intersectează se află pe o dreaptă comună.

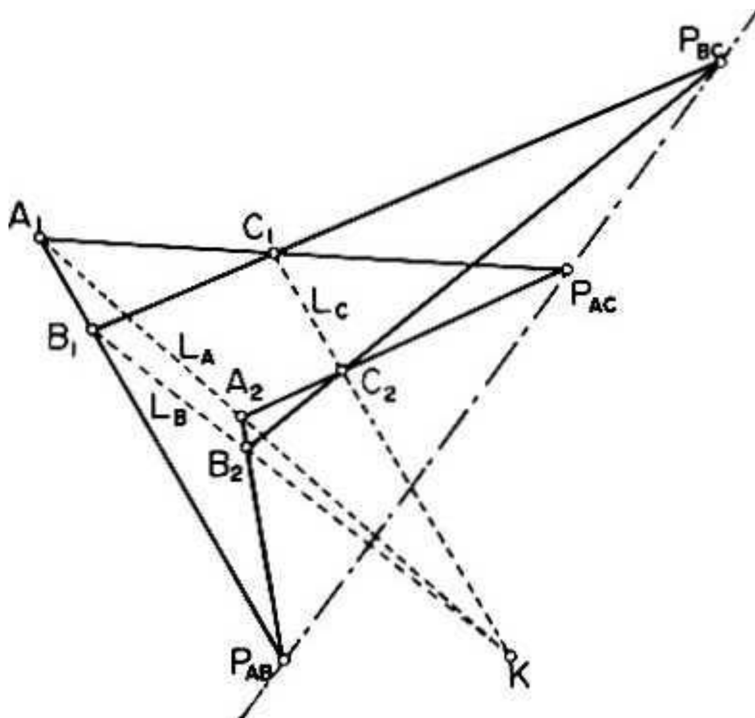


Figura 14.2 Ilustrarea teoremelor 14.1 și 14.2: punctele *p^, p^* **și** *Pgc* **sunt coliniare dacă și numai dacă dreptele** *L A . L t și L c* **trec printr-un punct comun .**

Figura 14.2 ilustrează teorema. Vezi (I4.F1J pentru o demonstrație. Teorema inversă poate fi exprimată imediat în termeni de dualitate.

Teorema 14.2: (Dualul teoremei 14.1.) Dacă punctele în care laturile (dreptele) respective ale două triunghiuri se intersectează se află pe o dreaptă comună, atunci liniile care unesc vârfurile (punctele) respective trec printr-un punct comun.

Principiul dualității poate fi folosit în grafică pentru a economisi software-ul. Astfel, aceeași procedură poate fi folosită pentru a găsi fie intersecțiunea a două linii, fie linia care unește două puncte. (În ambele cazuri evaluează minorii unui determinant 3X3.) Cititorul trebuie să țină cont de faptul că aplicarea dualității este posibilă numai atunci când sunt utilizate coordonate omogene. În plus, dualitatea este aplicabilă numai rezultatelor care descriu poziția relativă fără relații de distanță. De exemplu . luați în considerare următorul rezultat simplu: „Punctele bisectricei unghiului format din două drepte sunt echidistante de cele două drepte.” Nu este clar care ar trebui să fie dualul bisectrizei, dar deoarece trixul bisectrului este o linie, trebuie să fie un punct. Să-l numim *Q* astfel încât dualul afirmației de mai sus să devină: „Drecțiile care trec prin *Q* definit de

două puncte sunt echidistante de cele două puncte". Deoarece există o singură linie echidistantă de două puncte (normalul mediu la segmentul definit de acestea), ultima afirmație între ghilimele este greșită, f

14.4 PROBLEME DE SEGMENT DE LINIE

Rezolvarea multor probleme de vizibilitate, precum și a problemelor de tăiere . necesită calculul poziției relative a liniilor și punctelor. Vă prezentăm aici o colecție de rezultate care vor fi folosite în secțiunile ulterioare și în capitolul următor. În toate cazurile, presupunem că segmentele de linie sunt definite de punctele lor finale și că astfel de puncte finale sunt ordonate. Dacă segmentele fac parte dintr-un contur închis, atunci ordonarea poate fi în sensul acelor de ceasornic sau în sens invers acelor de ceasornic. Dacă nu se oferă nicio altă indicație despre comandă, atunci presupunem că este după cum urmează.

Ipoteza 14.1: Dacă nu se indică altfel, al) segmentele de linie sunt direcționate, cu săgețile îndreptate de jos în sus pentru segmentele neorizontale și de la stânga la dreapta pentru segmentele orizontale. Cu alte cuvinte, dacă (X|j' 1 .W|) sunt coordonatele punctului final listat primul și (^iJ'b^z) coordonatele punctului final listat al doilea, dacă yj/W] nu este egal cu *y^i.* atunci yi/w, este mai mic decât y^j. În cazul egalității (adică, un segment orizontal) avem Xj/wj mai mic decât x/w 2 (vezi figura

t Reader» sunt descurajați să încerce să se joace cu definiția* de „bisectrix", „echidistant". etc pentru a face ca dualitatea să funcționeze în acest exemplu. Un astfel de var poate fi cheltuit mai profitabil citind un text despre geometria proiectivă pentru a înțelege mai bine de ce documentele de dualitate nu se aplică relațiilor care implică metrica.

14.3). □

Definiția 14.1: Wc va spune că un punct *P se leagă de dreapta* unui segment de dreaptă *L* dacă se află la dreapta unui observator care merge de-a lungul liniei și merge de la primul punct de capăt la al doilea punct de capăt (vezi Figura 14.3). □

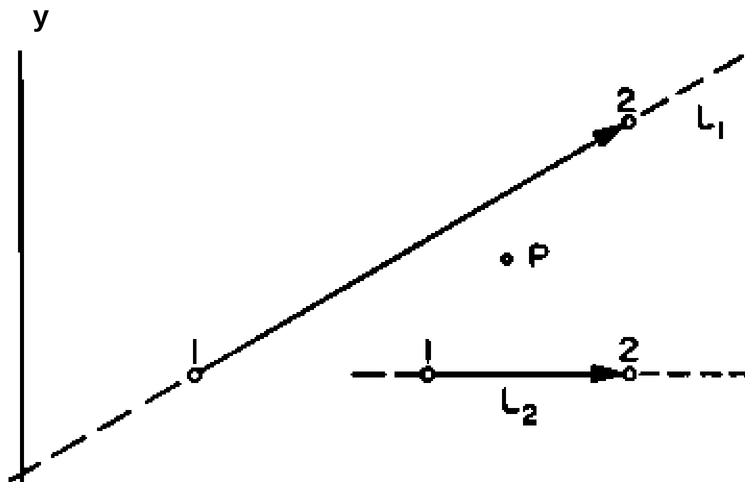


Figura **14.3 Ilustrarea notației implicite de ipoteza 14.1 și definițiile 14.1 și 14.2: punctul** *P* **se află la dreapta dreptei £i și la stânga dreptei** *L t . De asemenea,* **punctul** *P* **ascunde linia**

Definiția 14.2: Vom spune că un punct *P ascunde* o dreaptă *L* (sau un segment de dreaptă *L'),* dacă o dreaptă orizontală trasată prin *P* intersectează *L* (sau L9 într-un punct a cărui coordonată x este mai mică decât cea a *lui P* (vezi Figura 14.3). □

Relația „obscure" nu este valabilă dacă punctul *P* se află pe linia *L* și nu este definită dacă linia *L* este orizontală. În cazul unui segment de linie, relația nu este definită în următoarele trei cazuri: *P* se află deasupra punctului final superior. *P* se află sub punctul final inferior sau *P* aparține segmentului, iar segmentul este orizontal. Când orientarea unui segment este conform Ipotezei 14.1 și este definită relația „obscur", atunci ea este echivalentă cu relația „se află în dreapta lui".

14.4.1 Poziția unui punct față de o linie

Propoziția 14.1: Lei *X, Y* și *W* să fie coordonatele unui punct P și (xi.yi.W)) și (XJ.^.K'J) să fie punctele de capăt ale unui segment de dreaptă *L.* Dacă IP, W| și w*j sunt pozitive, atunci *P* se află în dreapta dreptei definite de segmentul *L* dacă și numai dacă

ArO^j-wj-j) + HWJXJ-X^J + ^(x^-PiXi) < 0 . (14,16)

□

Dovada: partea stângă a ecuației (14.16) este valoarea determinantului ecuației (14.14).

care este zero dacă P este un punct al dreptei care conține segmentul de dreaptă *L.* Deoarece determinantul este ecuația unei drepte, are un semn fix pentru punctele de pe fiecare parte a dreptei. Acolo deci. pentru a completa demonstrația, trebuie doar să verificăm că Ecuația (14.16) este valabilă pentru un punct din dreapta lui *L.* Dacă linia nu este orizontală, selectăm punctul

A"—x, + W| K =y, H' = w I .

Constatăm că după împărțirea cu *wfw 2* partea stângă devine

(ZL + „(ZL-ZZ) _ ZL^-ZL) + (ZhZi-ZiZL,
*1 w, *2 W| w, w 2 IV, w 2 w 2 W|

care este egal

Pi ri

Această cantitate este negativă din cauza ipotezei despre ordinea punctelor finale. Dacă linia este orizontală, repetăm un calcul similar selectând un punct cu .¥= x, iar F =y,-W|. □

Dacă se utilizează coordonate absolute, mai degrabă decât omogene, atunci Propunerea 14.1 este încă valabilă prin stabilirea IP - w, = w 2 - 1. În special . *P* va fi la dreapta dacă

(Pi“n) + Hxj-,) + UiPj-PiXj) < 0 (14,16')

Ecuațiile (14.16) și (14.16') pot fi scrise mai concis dacă folosim notația vectorială a ecuației (14.10). De asemenea, definim det(o.6.c) ca fiind determinantul unei matrice 3X3 cu coloanele a. *b,* și *c.* Această notație are avantajul suplimentar că este valabilă în coordonate absolute dacă se presupune că a treia coordonată a tuturor vectorilor este egală cu I. Atunci ecuația (14.16) (sau ecuația (14.16')) devine

det(PP It P 2) <0. (14,16")

Vectorii *P, P^* și *P 2* au definițiile evidente.

14.4.2 Intersecția segmentelor de linie

Putem folosi rezultatul de mai sus pentru a verifica dacă două segmente liniare se intersectează și, dacă se întâmplă, pentru a găsi intersecția. Dacă cele două segmente sunt definite de punctele *P\,P 2* și *P^P^* atunci cele două segmente se vor intersecta dacă și numai dacă coordonatele lui *P t* dau un semn diferit de coordonatele lui *P 2* când sunt substituite în ecuația dreptei care unește Pj și *P^.* O relație similară este valabilă atunci când rolurile lui *P t .P 2* și *P^P** sunt interschimbate (vezi Figura 14.4). Prin urmare, trebuie să calculați semnele următoarelor patru cantități:

5| - Xityj^-w^J +y,(w' > X4-x J *4) + w^x^“.?*««) (14.17a)

S 2 " XICFJ^-*^) +yj(*)X4-*)*4) + *2UjF4“)'y<4) (]4.I7b)

S 2 " *J(P1'*'2-*LF2) +yj(*l*2“*l*2) + "jUlPj-yi*!) (14.17c)

S A - X^Wj-WJ^) +y4(w ^2"X|* 2) + *4(Xiy l ->'l«2) (14.17d)

Condiția pentru intersecție este aceea că 5| și S 2 diferă în semn și același lucru trebuie să fie valabil și pentru S 2 și S4. Rețineți că această condiție este ceva mai puțin riguroasă decât cea folosită în Propoziția 14.1, deoarece nu ne pasă care sunt semnele atâta timp cât sunt diferite.

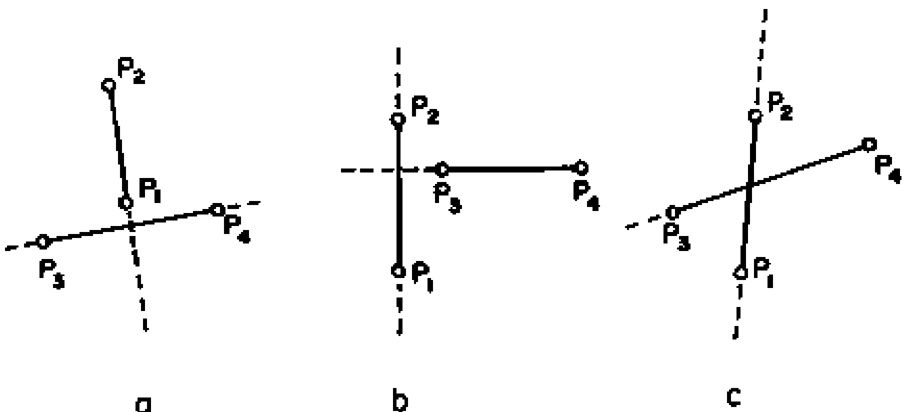


Figura 14.4 Ilustrarea semnelor ecuațiilor (14.17): (a) *S&* > 0 și *S^* < 0, (b) S^j < O și *SyS t* > 0, (c) S|S 2 < 0 și SjS* < 0

Dacă sunt îndeplinite condițiile pentru intersecție, atunci coordonatele punctului comun pot fi găsite prin rezolvarea următoarei perechi de ecuații liniare:

X^lWJ-WLPJ) + j(«'|X 2 -x 1 w 2) + wGr^-piXj) -0 (14.18a)

*CFJW4 – W1F4) + ^(w > x 4 -xjw 4) + H(x 1 p 4 -/ > x 4) - 0 (14.18b)

Rețineți că avem același număr de necunoscute și ecuații deoarece valoarea lui w este arbitrară.

Condițiile de intersecție pot fi scrise sub forma concisă notație după cum urmează.

S, - det(P | .P3.P4) . S2 = det (P2.PJ.P4) , S , S2 < 0 . (14.19a)

S)-det(Pj,P|.P2) . S 4 =■ det(P 4 ,P l .P 2). S^CO. (14.19b)

Ecuațiile (14.18) devin

det(PP„P 2) -0 (14.20a)

deUP.Pj.P^-O (14.20b)

Dacă unul sau mai multe dintre S/s sunt zero, atunci avem un caz singular și compararea semnelor nu poate fi efectuată.t Dacă S, este zero, atunci știm că P, este pe

linia definită de P_t și P_4. Pentru a afla dacă se află și pe segmentul definit de aceste două puncte trebuie să verificăm doar semnele lui S3 și S_4. Dacă unul dintre ele este zero, atunci știm că două dintre puncte coincid. Dacă ambele sunt zero, atunci cele patru puncte sunt coliniare. Suprapunerea segmentului poate fi verificată prin compararea dintre coordonatele x sau y.

14.4.3 **Poziția unui punct față de un poligon**

Dacă vârfurile unui poligon sunt ordonate în sensul acelor de ceasornic, atunci un punct va fi în interior dacă este întotdeauna la dreapta unui observator care traversează laturile în ordine. Dacă poligonul este convex, atunci cantitatea din partea stângă a ecuației (14.16) va fi negativă pentru toate laturile. În acest caz, soluția problemei este simplă.

Dacă poligonul nu este convex, atunci nu există o astfel de soluție simplă . Se poate arăta că orice punct din interiorul unui poligon neconvex II aparține unui poligon convex format din unele dintre laturile lui n și prelungirile lor (|3 PA], pp. 236-241). Cu toate acestea, găsirea acestora

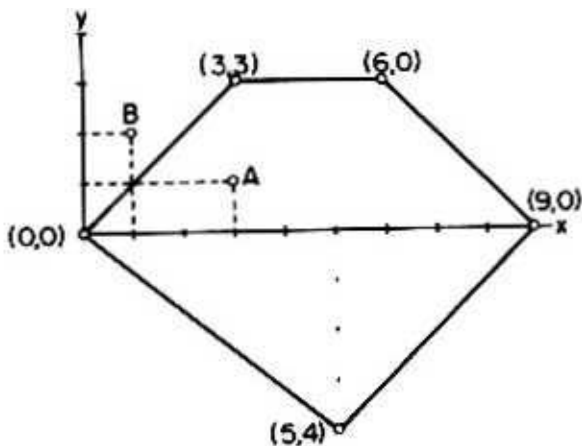
t Dacă cineva »alege patru puncte la întâmplare pe (planul euclidian il este foarte puțin probabil (hai trei dintre ele vor fi coliniare. Cu toate acestea, când ne mutăm la planul discret unde coordonatele sunt numere întregi mici (de exemplu, între 0 și $11) situația este diferită. Din cauza preciziei limitate utilizate în afișaje, astfel de cazuri singulare apar frecvent în practică.

poligoane este o problemă grea. În unele aplicații, acestea pot fi date în avans, de exemplu, dacă poligonul neconvex a fost construit ca o uniune de poligoane convexe. Efortul de a le găsi poate fi, de asemenea, justificat dacă intenționăm să verificăm poziția unui număr mare de puncte față de un poligon fix neconvex.

O altă soluție, valabilă pentru orice poligon, este să trasăm o dreaptă prin *P,* să găsiți intersecția acesteia cu toate laturile lui 11. și apoi să aplicați verificarea parității discutată în Secțiunea 8.1. Rezolvarea unui set de sisteme de ecuații necesită mult mai multă muncă decât verificarea semnelor unui set de inegalități. În plus, trebuie să numărăm efortul necesar pentru sortarea punctelor de intersecții. Astfel, metoda este utilă numai pentru poligoane neconvexe. (Desigur, situația este diferită dacă dorim să găsim toate punctele din interiorul poligonului.)

Exemplul 14.4: Fie vârfurile unui poligon convex (0,0), (3,3), (6,3), (9,0) și (5,-4) (vezi Figura 14.5). Fie *X* și *Y* coordonatele lui *P.* Cele cinci inegalități în acest caz sunt:

-3A + 3F < 0	sau	AT > F
3F - 9 < 0	sau	r < 3
3X + 3F - 27 < 0	sau	AT + F < 9
4AT -4F - 36 < 0	sau	*X* -*Y* <*9*
-4* - 5F < 0	sau	4* > -5F



Figare 14.5 Poligon utilizat în Exemplul 14.4. Pentru punctul A cu coordonatele (3.1) laturile din stânga primului set de inegalități au următoarele valori. —6. -6. -15. -28. și -17. Pentru punctul *B* cu coordonatele (1,2), partea stângă a primei calități ine este egală cu 3.

14.4.4 Umbra segmentului

Rezolvarea anumitor probleme de vizibilitate necesită examinarea poziţiei relative a segmentelor liniare.

Definiția 14.3: Se spune că segmentul *a umbră sau ascunde segmentul b* dacă nu intersectează segmentul *b* și dacă cel puțin unul dintre punctele sale îl întunecă *pe b.* □

Figura 14.6 prezintă câteva exemple de segmente și, de asemenea, listează care segmente le umbră pe altele conform definiției de mai sus. Rețineți că relația „segmentul *a* nuanțește segmentul *b"* nu este tranzitivă și nici negația relației „a nu nuanțește *b".* Prin urmare, relația de „umbrire" nu introduce o ordonare parțială, cu excepția afirmațiilor cu efect opus în literatură (vezi Notele Bibliografice).

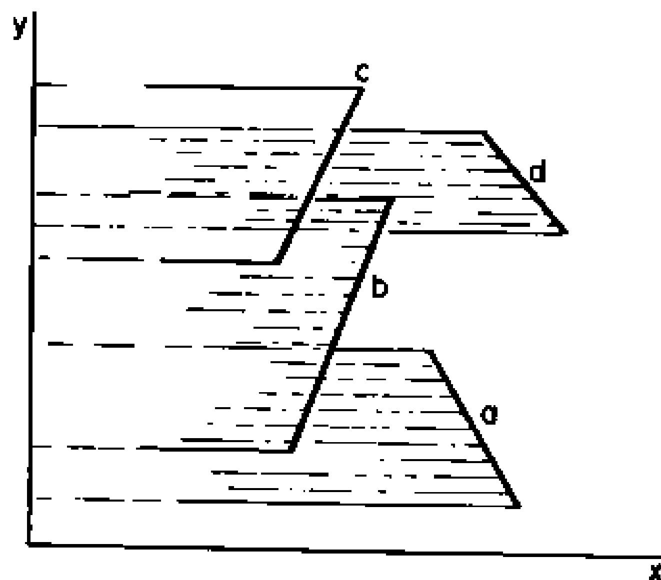


Figura 14.6 **Ilustrarea umbririi segmentului:** *a* **nuanțe** *b. b* **nuanțe c, dar a nu nuanță** *c. d* **nuanțează atât** *b,* **cât și c, dar nu umbrează** *a.*

Următoarea propoziție este o consecință directă a rezultatelor secțiunilor anterioare.

Propunerea 14.2: Segmentul *a* umbrește segmentul *b* dacă și numai dacă sunt - valabile următoarele trei condiții:

Jt 2Pi (14.21a)

yî^y? (14.21b)

*?(yi -yj)-yi(*i - ^)+xH *-yM* <o (i4.2ic)

□

14.5 NOTE BIBLIOGRAFICE

Majoritatea textelor despre grafică citate mai devreme discută despre crearea unor afișaje bidimensionale simple. Tratamentul detaliat al coordonatelor omogene poate fi găsit în toate textele despre geometria proiectivă, cum ar fi 114.FI), (I4.MA), [14.PE], [14.RO] și [14.TO]. Formulele pentru transformarea coordonatelor după rotație pot fi găsite în majoritatea textelor de geometrie analitică, cum ar fi (14.LE). Teoremele 14.1 și 14.2 se datorează lui Desargues (1593-1661). Geometria proiectivă este studiul proprietăților geometrice care sunt invariante în cadrul transformărilor liniare, cum ar fi înmulțirea cu matricele din Secțiunea 14.2. Originile sale datează din secolul al IV-lea d.Hr. (Pappus din Alexandria), dar a devenit un subiect matur abia în secolul al XIX-lea. În mod clar, merită puțină atenție din partea oricui este serios interesat de graficele . Pentru o discuție despre proprietățile relației de umbră și subiecte similare vezi [14.GY1.

O problemă provocatoare în grafică este implementarea unora dintre transformările geometrice pe planul discret și nu pe cel analog. Acest lucru ar putea economisi atât calculul, cât și comunicarea între procesoare. Se pare, totuși, că există anumite dificultăți insurmontabile cu o astfel de abordare. Cititorii se pot convinge că acest lucru este într-adevăr așa încercând să scrie un program pentru rotații pe planul discret.

14.6 LITERATURA RELEVANTA

[14.FI) **Fishback, WT** *Geometrie proiectivă și euclidiană.* **New York: J. Wiley, 1969.**

[I4.GY1 Guibas, LJ și Yao, FF „Despre traducerea unui set de dreptunghiuri", *Proc. Al doisprezecelea ACM Symp. despre Teoria Calculului.* **Los Angeles, Calif.. (aprilie 1980), p. 154-160.**

[I4.LE] Lehmann, CH *Geometrie analitică.* **New York: J. Wiley, 1942. (tipărire 12 inchi*, 1961.)**

11 4. MA) Maxwell, EA *Geometrie proiectivă plană Bazată pe utilizarea coordonatelor omogene generale.* **Cambridge: University Press, 1957 (retipărire a primei ediții din 1946).**

[14.PE) Pcdoe, D. ^FT *Introduction to Projective Geometry,* **New York: Mac Millan, 1963.**

[14.ROJ Rosenbaum, RA *Introducere în geometria proiectivă și algebra modernă.* **Reading, Mass: Addison-Wesley, 1963.**

[I4.TOJ Todd, JA *Geometrie proiectivă și analitică.* **New York: Pitman. 1948.**

14.7. PROBLEME

14.8. Scrieți un program care să afișeze un hexagon în dimensiuni succesive mai mici până când acesta se micșorează la un singur pixel.

14.2. Fie C o curbă definită de o succesiune de puncte {;w3r. Este posibil să afișați C printr-o secvență de instrucțiuni *vec* , dar pentru a produce efectul de grosime, este posibil să dorim să folosim un „pen”. Considerăm aici cazul unui stilou cu vârf pătrat: curba va fi afișată printr-o succesiune de pătrate, fiecare centrat la unul dintre punctele *x*^ cu o pereche de laturi ale pătratului fiind paralele cu dreapta definită de punctele jr,-|.)7-| și *x* *lt* *y,.* Scrieți un program care implementează un astfel de afișaj.

14.3. (Extinderea problemei anterioare.) Vârful pătrat nu este foarte estetic. Cel mai bine este să folosiți o elipsă pentru un vârf. Scrieți un program care execută această sarcină cu constrângerea că axa mică a elipsei este întotdeauna paralelă cu vectorul care unește două puncte succesive.

14.4. Adăugați în editorul Problemei 10.6 o caracteristică pentru rutarea, mărirea și/sau translatarea unei curbe.

14.5. Scrieți un program care să afișeze planetele care se mișcă în jurul soarelui. Poate doriți să utilizați o scară de distorsionare și să alegeți o săptămână pământească ca interval de timp dintre reînnoirea afișajului. Simplificați problema făcând ca traiectoriile planetelor să fie cercuri care se află toate pe același plan, astfel încât să puteți utiliza ecuațiile din secțiunea 14.2. Puteți alege poligoane diferite pentru fiecare planetă (de exemplu, pătrat pentru Pământ, hexagon pentru Marte etc.), sau poligoane similare, dar de culoare diferită, sau orice altă combinație, în funcție de tipul de afișaj la care aveți acces.

14.6. Scrieți o procedură pentru găsirea intersecției a două drepte în coordonate omogene și utilizați principiul dualității pentru a avea aceeași procedură să găsească dreapta definită de două puncte.

14.7. Scrieți un program care execută o buclă infinită cu următorii pași:

(1) Solicitați utilizatorului un cod de culoare și patru coordonate. (Puteți dori să rezervați un cod de culoare ca steag pentru o ieșire plină de grație.)

(2) Afișați segmentul dintre cele două puncte definite de coordonatele date.

(3) Schimbați culoarea tuturor segmentelor afișate astfel încât, dacă

segmentul a *ascunde* b *, b* să fie afișat în culoarea a.

(Dacă nu aveți acces la un afișaj grafic color, afișați o literă lângă fiecare segment și utilizați acele litere în loc de codul de culoare.)

Capitolul 15

DECUPARE POLIGON

15.1 INTRODUCERE

Termenul de tăiere este folosit pentru a descrie procesul de a afla dacă o linie (sau poligon) este intersectată de un alt poligon. O aplicație majoră a tăierii este de a decide dacă anumite construcții se încadrează în fereastra de afișare. Ar trebui să subliniem aici că nu se poate baza pe dispozitivul de afișare pentru a realiza tăierea din două motive. În primul rând, zona de afișare poate fi doar un subset al ecranului de afișare (de exemplu, un par al ecranului poate fi rezervat pentru afișarea textului). În al doilea rând, nu există o modalitate uniformă prin care sunt afișate punctele cu coordonate nevalide. Destul de des, un dispozitiv de afișare ignoră biții de ordin înalt, astfel încât construcțiile se înfășoară în jurul ecranului. Prin urmare, este nevoie de - algoritmi pentru a rezolva această problemă în mod explicit.

O altă utilizare a tăierii este în rezolvarea problemelor de vizibilitate. Pentru a decide dacă un obiect îl ascunde pe altul, trebuie să găsești mai întâi dacă două obiecte se intersectează. O a treia utilizare care a fost explorată foarte puțin este descompunerea unui poligon neconvex în subpoligoane convexe. Acest lucru poate fi util nu numai în grafică (vezi Secțiunea 14.4.3), ci și în recunoașterea modelelor. Acolo subseturile convexe ale unui poligon pot fi folosite ca elemente primitive pentru descrierea formei poligonului original (vezi Notele Bibliografice).

Vom folosi rezultatele din Secțiunea 14.4 pentru a proiecta diverși algoritmi de tăiere. Secțiunea 15.2 descrie tăierea unui segment de linie de către un poligon convex, în timp ce secțiunea 15.3 rezolvă aceeași problemă pentru
caz practic important când poligonul este un dreptunghi cu laturile paralele cu axele de coordonate. Secțiunea 15.4 tratează decuparea unui poligon arbitrar de către o linie, iar Secțiunea 15.5 cu tăierea unui poligon de un alt poligon. În cele din urmă, Secțiunea

15.6 examinează problema eficienței de calcul pentru unele probleme geometrice.

15.2 DETUNEREA UNUI SEGMENT DE LINIE DE UN POLIGON CONVEX

Această problemă este de generalitate intermediară între extremele tăierii unei linii printr-un dreptunghi regulat (vezi secțiunea următoare) și tăierea unei linii printr-un poligon neconvex.

Problemă: Dacă^|,^2. '"^» arc vârfurile unui poligon convex II pe plan, găsiți partea unui segment de dreaptă *L* (definită prin punctele sale terminale *Pt* și P_2) care se află în interiorul poligonului □

Rețineți că soluția naivă, care verifică dacă ambele puncte *P* și P_2 sunt în interiorul poligonului, este incorectă deoarece un segment de linie poate intersecta un poligon convex chiar dacă ambele puncte terminale sunt în afara poligonului. Următoarea soluție, bazată pe materialul din secțiunea 14.4, este posibilă. Mai întâi evaluați cele *n* mărimi

Si - det(^,P|,P_2) - *Xi(yiw* $_2$ - w^)

+ FJwpXj-x,w $_2$) + ^(x^jy,x $_2$) 1-1.2. ■■• n (15,1)

unde *A* $_{\{}$ denotă vectorul de coordonate (A^Pi) al vârfului A_t al poligonului și *Pj* (J —1,2) vectorul de coordonate *(XJ^J)* al punctelor de capăt ale dreptei *Pj.* Rezultatele calculului vor cădea într-unul din următoarele cazuri.

(1) Toate V-urile sunt diferite de zero și au același semn. Atunci toate vârfurile poligonului se află pe aceeași parte a lui *L* și nicio parte a liniei nu se află în interiorul poligonului.

(2) Una dintre ele este zero și toate celelalte au același semn. Apoi linia trece printr-unul dintre vârfuri, dar toate celelalte vârfuri sunt de aceeași parte a liniei. Acest caz este echivalent cu (1).

(3) Două dintre ele sunt zero și toate celelalte au același semn. Deoarece II este convex, acest lucru se poate întâmpla numai atunci când două vârfuri succesive se află pe *L.* Atunci avem cazul singular discutat în Secțiunea 14.4.2: o latură a poligonului se află de-a lungul aceleiași drepte cu segmentul *L.*

(4) Una sau două dintre ele sunt zero, iar celelalte au semne diferite. Aceasta înseamnă că *L* intersectează II în timp ce trece prin unul sau două dintre vârfurile sale. Deoarece n este convex vor exista doar două modificări ale semnelor și vom analiza acest caz împreună cu următorul.

(5) Toate S_t*-urile* sunt diferite de zero, dar au semne diferite. Apoi vor avea loc două schimbări de semn pe măsură ce unul merge de-a lungul perimetrului poligonului. Vom folosi *A,. A* $_{J+i}$ și A_k, *A^* pentru a desemna perechile unde are loc schimbarea. S_t poate fi zero într-unul dintre vârfurile uneia sau ambelor perechi dacă apare cazul (4). Li va desemna segmentul de linie care unește prima pereche și L_2 segmentul de linie care unește a doua pereche. Rețineți că ce pereche va fi numită prima și care a doua este arbitrară, iar notația particulară este lipsită de importanță atâta timp cât suntem

consecvenți.

Odată ce aflăm că linia care conține *L* intersectează poligonul (ultimele două cazuri) trebuie să verificăm poziția punctelor P (și P_2 față de liniile L, și L_2, adică să găsim semnul celor patru cantități) .

l/rdeKP^A^-x^^^-^W

+ y,(^A}.H “ Aj^+.) + *i(^r /+ , - *YjX^* (15.2a)

(locația lui P, în raport cu segmentul L,)

U 2 - deKP^pAz-H) „ *x* 2 *(Yj^* *f+i* - H^-H)

+ ^(^,-^^1) + *w* 2 *{XJY^ - Y t X^)* (15.2b) (locația lui P 2 față de segmentul L|)

U 2 “ det(P|,Ak,An.|) = xj(k* W^+i — *W k* K* +1)

+ *y l (W k X k + l -X k W k ^) + w { (X k Y k ^ - Y k X k ^)* (15.2c) (locația lui P| în raport cu segmentul £ 2)

u *4* - det(p 2 ,^ A* +1) - x 2 (r* FP* +1 - ^r* +1)

+/:(^ z *^*i ¯ A*1F t+l) + wjfAkKk+i — y*X 4+l) (15.2d)

(locația lui P_2 în raport cu segmentul £ 2)

Dacă direcțiile laturilor poligonului sunt stabilite în sensul acelor de ceasornic, semnul lui U_t în ecuațiile (15.2) este negativ atunci când un punct se află în interiorul poligonului. U_t va fi zero dacă o latură conține unul dintre punctele P sau P_2. Omitem discuția unor astfel de cazuri singulare și lăsăm modificarea necesară a algoritmului 15.1 ca exercițiu (Problema 15.1). Figura 15.1 prezintă diferite poziții relative ale punctelor Pj și P_2 în raport cu cele două linii, iar Tabelul 15.1 listează semnele lui *U { și* calculul punctelor finale ale segmentului afișat. Acest tabel și algoritmul folosesc următoarea notație: 0 și g_2 arc punctele finale ale segmentului de linie care urmează să fie desenat. L n£ z (i - 1. 2) denotă punctul în care segmentele *L* și L_t se intersectează .

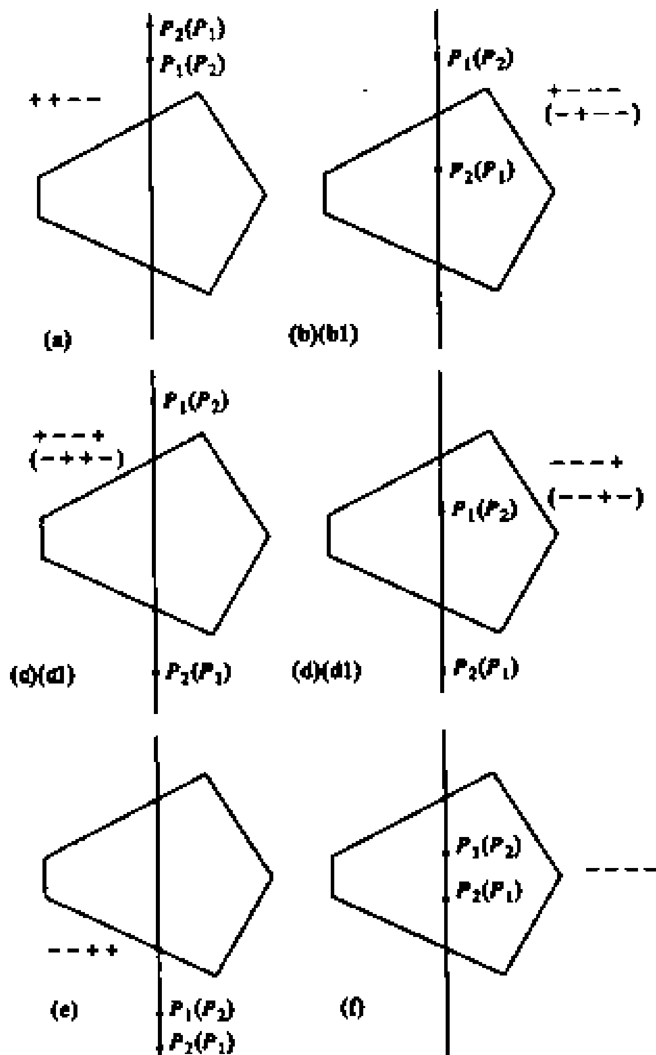


Figura 15.1 **Decuparea unui segment de linie de către un poligon. Sunt prezentate cele șase poziții posibile ale unui segment în raport cu un poligon. Trei dintre ele au o etichetare dublă, astfel încât avem un total de nouă cazuri, așa cum se arată în Tabelul 15.1.**

Teoretic avem șaisprezece posibilități nesingulare (semnele a patru variabile), dar din cauza constrângerilor geometrice avem doar nouă cazuri. Într-adevăr, este imposibil să aveți toate *U, sunt* pozitive deoarece semiplanurile pozitive ale celor două laturi sunt în afara poligonului. În mod similar, este imposibil să existe trei semne pozitive. Aceasta rămâne unsprezece cazuri. Configurația - + -+ înseamnă că P, se află în interiorul poligonului în timp ce P_2 este deasupra dreptei *AjAj*+\ și sub dreapta X t^ + i, o imposibilitate. Se poate face un argument simetric pentru configurația 4 F—, astfel încât să rămână cele nouă cazuri prezentate în Tabelul 15.1. Rețineți că în ceea ce privește configurațiile geometrice reale avem doar șase cazuri, cele prezentate în Figura 15.1. Trei dintre cazuri pot avea una dintre cele două etichete posibile. Acestea sunt cazurile *b. c* și *d* în Figura 15.1, unde a doua etichetare este prezentată în paranteze. Dacă presupunem că Pi este sub P2, ne putem limita atenția la doar șase cazuri, dar dorim să evităm această presupunere în cazul în care P|P2 este o latură a unui poligon în care ordonarea punctelor terminale nu este determinată de coordonatele lor y.

Tabelul 15.1

Poziția unui segment de linie față de un poligon

Figura	^1	^7		^4	21	2i
(o)	+	+			invizibil	
(b)	+	—	-	—	*LCtLi*	*Pi*
(b)	—	+	—	-	*Pi*	*LnL t*
(C)	+	—	•	+	*LKLi*	*LC\L 2*
(C)	—	+	+	—	*LOL 2*	*LC\L }*
(d)	—	-	—	4-	*Pi*	*L* AL 2
(d)	—		+	—	*LDL 2*	^2
(e)	—	-	+	+	invizibil	
	—	-	—	-		

Algoritmul 15.1 verifică semnele mărimilor din Ecuația (15.1) și Ecuațiile (15.2) și implementează deciziile enumerate în Tabelul 15.1. Prima parte examinează locația dreptei în raport cu vârfurile poligonului, în timp ce a doua examinează poziția extremităților dreptei. Variabilele v și u sunt setate egale cu semnele lui 5 (, așa cum a fost calculat din ecuația (15.1), pentru vârfurile succesive ale poligonului. Dacă *u* și v diferă, înseamnă că cele două vârfuri se află pe laturile opuse ale dreptei care conține *L.* Coordonatele acestor vârfuri sunt salvate în pasul 5. Deoarece *astfel* de numărători se presupune că numărul de conexiuni al poligonului se presupune. valoarea sa trebuie să fie zero sau două Dacă se constată că este zero la sfârșitul buclei pașilor 3-.

b. atunci știm că linia care conține *L* nu intersectează poligonul și am terminat (pasul 7). În caz contrar, continuăm cu pașii 8-17.

Pasul 9 evaluează ecuațiile (15.2), iar pașii rămași efectuează verificările din Tabelul 15.1. Cazurile (a) și (e) în care segmentul se află în afara poligonului sunt verificate la pasul 10. Dacă verificarea pasului 10 eșuează, atunci o inspecție a figurii 15.1 și a tabelului 15.1 arată că dacă L/| și t/j au semne opuse, linia £ va intersecta unul

dintre £ (sau £ 2 . În mod similar pentru U_2 și *U4.* Aceste verificări se fac la pașii 14 și 15. mai întâi pentru *U* l și *U* j, apoi pentru U_2 și *U 4.*

Din cauza dualității, acest algoritm de tăiere poate fi folosit și pentru a rezolva următoarea problemă:

Problemă: Având în vedere *n* drepte care sunt laturile unui poligon convex, aflați dacă un punct dat ca intersecția a două drepte se află în acel poligon. □

Soluție: Aplicați dualul algoritmului 15.1. □

Algoritmul 15.1 Decuparea unui segment de linie de un poligon convex

Notație: **Tabloul** *A* **are dimensiunea nX3 și conține tripletele din coordonatele homogene ale celor** *n* **vârfuri ale poligonului. Coordonatele punctelor terminale ale segmentului de linie £ arc stocate în tabloul 2X2** *P.* **Matricele** *B* **și** *C* **au dimensiunea 2X3-** *B* **este folosit pentru a stoca primul dintr-o pereche de vârfuri situate de fiecare parte a liniei care conține** *L', C* **conține al doilea membru al unor astfel de perechi.** *S* **și** *U* **sunt definite în ecuația (15.1) și ecuațiile (15.2). Algoritmul găsește** Q_t **și** Q_2 *, punctele* **finale ale segmentului conținut în poligon.**

1. Setați contorul *K* la zero.

2. Setați v egal cu semnul lui Sj din ecuația (15.1) care este evaluat folosind >4(1,1) pentru I h >4(1,2) pentru Kj etc.

3. Pentru 1 ■■ 2 la n face: ...

ÎNCEPE.

4. Setați *u* egal cu semnul lui *S* (în ecuația (15.1) care este evaluată folosind 4(/.I) pentru^, 4(i,2) pentru *Y* *It* etc.

5. Dacă v nu este egal cu u, atunci creșteți *K,* setați *B(K^)* la 4(i-IJ), *C(KJ) tQA(iJ)* și *v =u.*

6. **Dacă** *K* este egal cu 2, **atunci rupeți** din buclă. **Sfârşit.**

7. **Dacă** *K* **este egal cu zero, atunci ieșiți.**

8. **Altfel face:**

ÎNCEPE.

9. Configurați ecuațiile (15.2) folosind tabloul B(l?) pentru punctul *A^.* tabloul C(l,*) pentru >4 /+h tabloul B(2?) pentru A_k, iar tabloul C(2,*) pentru 4* +b Apoi se evaluează expresiile pentru l/| la *U4.*

10. Dacă U_x și *U^* au același semn și acest semn este opus celui al lui U_2 și U 41 **apoi iesi.**

11. **Altfel face:**

ÎNCEPE.

12. **Pentru** m = 1 la 2 **faceți:**

ÎNCEPE.

13. **Dacă** U_m și U_{m+2} au semne opuse, **atunci procedați:**

ÎNCEPE.

14. **Dacă** semnul lui U_m este *minus,* **atunci** setați Q_m la

intersectia segmentelor de dreapta £ si

15. **Altfel,** setați *Q^* la intersecția lui £ și

Sfârşit.

16. Altfel setați *Q„,* egal cu P_m.

Sfârşit.

Sfârşit.

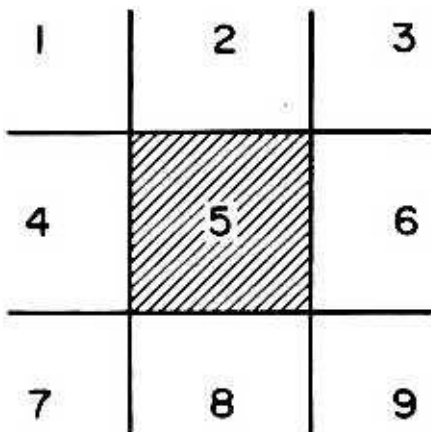
Sfârşit.

17. Afișați segmentul dintre punctele Ci și *Qi-*

18. **Sfârșitul algoritmului.**

19. DREPT REGULARE

Un caz special al algoritmului de tăiere apare atunci când n — 4 și poligonul este un dreptunghi cu laturile paralele cu axele de coordonate. Vom numi un astfel de dreptunghi *regulat.* Este posibil să se efectueze câteva verificări simple pentru locațiile lui P_x și P_2 în raport cu liniile verticale și orizontale care delimitează dreptunghiul. În Figura 15.2, regiunea 5 este fereastra de vizualizare. Dacă ambele puncte se află într-o singură regiune, cu excepția 5, sau dacă ambele se află într-unul dintre grupurile (1,4.7), (1,2,3), (3,6,9) sau (7,8,9), segmentul nu este vizibil. Este vizibil atunci când ambele puncte se află în 5. Trebuie să examinăm problema vizibilității în detaliu pentru toate celelalte locații ale punctelor finale.



Figare 15.2 Decuparea cu o fereastră dreptunghiulară

Acest caz special este de o importanță practică suficientă pentru a justifica un algoritm special de tăiere. Este listat ca algoritmul 15.2. Pasul I verifică dacă segmentul se află în întregime într-unul din cele patru grupuri de trei blocuri, fiecare prezentat în Figura 15.2: (1,4,7), (3,6,9), (7,8,9) sau (1,2,3). Apoi întregul segment se află în afara ferestrei și nu este nevoie de procesare ulterioară. Pasul 2 se asigură că etichetarea punctelor finale este în acord cu ipoteza 14.1. Pasul 3 (inclusiv 3a-3c) tratează o linie orizontală, iar pasul 4 (inclusiv 4a și 4b) cu o linie verticală. Pașii 5-11 tratează cazul

general. Am fi putut omite pașii l, 3 și 4 fără a provoca erori, dar în detrimentul unui calcul crescut. Acești pași elimină rapid cazurile în care nu este necesară aplicarea criteriului general de intersecție a liniilor. Acesta este un exemplu în care o creștere a lungimii algoritmului scade timpul mediu de execuție.

Pasul 5 evaluează semnele celor patru colțuri în raport cu linia. În special, l/, corespunde colțului din stânga jos *(XM/N>Y M/N),* ^2 către *(^MIN^MAX^* ^i colțului din dreapta sus *(XM^YHAX)* și *U 4* către *(X MAXt Y MIN).* Condiția pasului 7 este adevărată atunci când linia intersectează limita din stânga a ferestrei. linie pentru toate calculele ulterioare Rețineți că este imposibil să fie valabilă condiția ambilor pași (adică atât Xi cât și x 2 mai mici decât *X MIN),* algoritmul nu ar fi trecut de pasul 1. Condiția pasului 8 este adevărată când linia intersectează.

limita superioară a ferestrei. Pasul 8b calculează intersecția. Rețineți că, din cauza Ipotezei 14.1, trebuie să verificăm locația unui singur punct final. Condiția pasului 9 este adevărată atunci când linia intersectează limita dreaptă a ferestrei și cea a pasului 10 este adevărată când linia intersectează limita de jos a ferestrei. Pașii 9b, 9c și 10b calculează intersecțiile și le folosesc pentru a înlocui punctele finale.

Rețineți că doar două dintre cele patru condiții ale pașilor 7, 8, 9 și 10 pot fi adevărate. Prin urmare, este posibil să se accelereze algoritmul prin sărirea peste verificările semnelor rămase de îndată ce *j* atinge valoarea 2. O altă observație este că, dacă segmentul de linie este conținut în întregime în fereastră, niciuna dintre verificările 7b, 7c, 8b, 9b, 9c sau 10b nu va fi adevărată și singurul calcul netrivial al pasului 5 va fi acel calcul netrivial.

Algoritmul 15.2 Decuparea unui segment de linie printr-un dreptunghi regulat

Notație: Extremitățile liniei sunt (x^J și (XJ^J). Dreptunghiul este de la *Xym* la *X»M* și *YK IN* la *Y UAX- draw* este procedura care afișează linia. *dy, dx, dxy, qx, qX. qy* și *qY* sunt mărimi auxiliare definite la pasul 5.

1. Dacă (X| și x 2 sunt ambele mai mici decât *X MW)* sau (x (și x 2 sunt ambele mai mari decât *X UAX }* sau O'i și y 2 sunt ambele mai mici decât *Y UIN }* sau *(y t* și j 2) sunt ambele mai mari decât *Y UAX),* **apoi revin.**
2. **Dacă** ^i **depășește** *y 2 ,* **atunci** schimbați punctele finale.
3. **Dacă** *y }* este egal cu *y* 2 **, atunci**

ÎNCEPE.

3a. **Dacă** Xi depăşeşte x 2 . **apoi** schimbă punctele finale.

3b. **Dacă** x, este mai mic decât *X UIN ,* **atunci** setați X| *„ X MIN .* **Dacă x 2** este mai mare decât *XMM,* **apoi** setați x 2 - *X MAX .*

3c. d><2»v(x l ,^|,x 2 ^ 2) și **întoarcere.**

Sfârşit.

4. Dacă X| este egal cu x 2 , atunci

ÎNCEPE.

4a. **Dacă** yi este mai mic decât *Y MIN ,* **atunci** setați yi - *Y MW .* **Dacă** y 2 este mai mare

decât Y_{UAX}, **atunci** setați/ $_2$ - Y^.

4b. draw-fxi j^^j^) și d **return.**

Sfârşit.

5. Set

*dy-yi -yi^* = *i - x_2 \dxy „Xi^-yi^

$qx^B X_{MW} dy\backslash qX = Xiwdy, qy - Y$^ -dx-, $qY - Y_{MAX}$ dx. U^ — $qx — qy + dxy. U_2$

~ $qx \sim qY + dxy, U_3 — qX —qY +dxy. U_4 = qX-qy +dxy.$

6. Setați *j* la zero.

7. Dacă L/1 și U_2 diferă ca semn, atunci Începe.

7a. Crește *j.*
7b. Dacă x, este mai mic decât X_{MIN}, atunci setați
Xi " *(Xwdy + dxy)/dx, x^ — X^.* Actualizați *dx, dy* și *dxy.*

7c. Dacă X] este mai mic decât X_{UiN}, atunci setați
3*2 " *($X_{M/N}$ dy +dxy)/dx* ; xj — X_{MIN}. Actualizați *dx, dy* și *dxy.*
Sfârşit.

8. Dacă U_2 și U_y diferă ca semn, atunci Începe.

8a. Crește *j.*
8b. Dacă j^ este mai mare decât K^^. apoi
**2 " WMAX'dx -dxy)/dy;y* $_2$ - *Y^AX*- Actualizați *dx,dy* și *dxy.*
Sfârşit.

9. Dacă Uj și U_4 diferă ca semn, atunci Începe.

9a. Crește *j.*
9b. Dacă X| este mai mare decât X_{UAX}, apoi setat
>*i " <X_{MAX} *dy +dxy)/dx;* X| ^ X_{UAX}. Actualizați dx, *dy* și *dxy.*

9c. Dacă x2 este mai mare decât *X^^*, atunci setați
/2 " (X_{UAX} dy +dxy)/dx; x_2 - *Xiw* Actualizați *dx, dy* și *dxy.*
Sfârşit.

10.
Dacă *Ui* și $U_}$ diferă ca semn, atunci Începe.
10a. Crește *j.*
10b. Dacă ^i este mai mic decât Y_{UiN}, atunci setați
*I „ *(Y_{MW} dx ~dxy)/dy,y* $_}$ - Y_{U1N} Actualizează dx, *dy* și *dxy .*
Sfârşit.

II. Dacă J este mai mare decât zero, atunci dron^x^ ix^)-
12. Reveni
13. Sfârșitul algoritmului.

20. 4 TUNTAREA UNUI POLIGON ARBITRAR CU O LINIE

Pentru a afișa un poligon care nu este neapărat convex peste o fereastră care este un dreptunghi obișnuit, putem fie tăia fiecare dintre laturile sale în raport cu fereastră (Algoritmul 15.2), fie decupăm întreg poligonul în raport cu fiecare dintre laturile ferestrei. Algoritmul 15.3 rezolvă această din urmă problemă. Poate fi folosit și ca algoritm de bază pentru găsirea intersecției a două poligoane arbitrare.

Algoritmul calculează l/ $_($ și t/ $_2$, așa cum este dat de ecuațiile (15.2) pentru toate

vârfurile. (Rețineți că, pe măsură ce avansăm de la o pereche la alta, noul *Ui* este egal cu vechiul *U*, astfel încât calculul trebuie făcut o singură dată la fiecare vârf.) Se presupune că linia este orientată conform Ipotezei 14.1 și care nu sunt vizibile sunt acele puncte care nu sunt vizibile. 14.2). Astfel, ***dacă*** U *t este pozitivă ,* vârful *este* vizibil .

Teoretic, este imposibil să existe *, egal cu zero în pasul 5, deoarece acest lucru ar implica că segmentul este paralel cu dreapta și, prin urmare, Ui și *U 2* au același semn, o contradicție. Tabelul 15.2 rezumă acțiunile întreprinse de algoritm în funcție de semnele lui *U* l și *U2 .*

Tabelul 15.2 Poziția unui poligon față de o dreaptă

^1		Sutus	Pasul algoritmului
+	+	toate vizibile	4
+	0	alt vizibil	4
+	—	parțial vizibil	7
0	+	toate vizibile	4
0	0	invizibil	3
0	—	parțial vizibil	7
—	+	parțial vizibil	6
—	0	invizibil	3
—	—	invizibil	3

Când ambele puncte sunt pe linie, algoritmul tratează segmentul ca fiind invizibil. De fapt, poligonul desenat va fi deschis acolo unde linia îl intersectează. așa cum se arată în Figura 15-3- Acest lucru poate fi nedorit în anumite aplicații, dar nu este dificil să se modifice algoritmul astfel încât intersecțiile să fie unite prin linii (vezi problema 15.4).

Algoritmul 15.3 Decuparea unui poligon de o linie

Notație: intrarea în algoritm este coeficienții *dY, dX* și *D* ai ecuației dreptei:

dY x -dX y + D -0

iar coordonatele vârfurilor poligonului *x,, y,, i* — I, 2. • • - *N.* Rezultatul este afișarea poligonului tăiat care nu este ascuns de linie. *dy. dx,* și *dd* arc coeficienții dreptei definite de o latură de poligon și arc calculat la pasul 5. Coordonatele omogene ale intersecției se găsesc conform metodei din Secțiunea 14.3.2.

1. *Set U l =dYx 1 -dXy l + D.*
2. **Pentru** *i* de la 2 la A' **faceți** pașii 3-8.

    ÎNCEPE.

3. Setați U 2 = *dYx, - dXy t + D.* **Dacă** atât *U {* cât și t/ 2 nu sunt pozitive, **atunci** nu faceți nimic.

    Altfel

    ÎNCEPE.

4. **Dacă** ambele l/| și *U^* sunt nenegative , **apoi** *desenați* vector de la {x,-,^-,) la (x,^).

Altfel

ÎNCEPE.

5. Set dy = ^._| — *y* *it* *dx* — x (_| — x /t și

dd = x^-fy, — *y* *t* -*i'x* *lt* și apoi se calculează x fl - - *{dxD -dX D),*

y *a* = - *(dy D - dY D)* și *w* *a* = -*(dydX -dY dx).*

dacă *w* *a* nu este zero, **atunci** setați *x^* ^ *x* *o* /*w* *a* ^ și *y^* = *y* *a*

/^ *a* -

Altfel, ridicați indicatorul de eroare și **ieșiți.**

6. **Dacă** U| este negativ, **apoi** *trageți* vectorul din

{x *t* „*y* *tltr* *)* la (x,^).

Altfel

ÎNCEPE.

?■ I^i este nenegativ şi *U* *2* >s pozitiv

Desenați vectorul de la x,-,^-, la *x^y^.* **Sfârşit.**

Sfârşit.

Sfârşit.

8. Set(/| = U 2 .

Sfârşit.

9. **Sfârșitul algoritmului.**

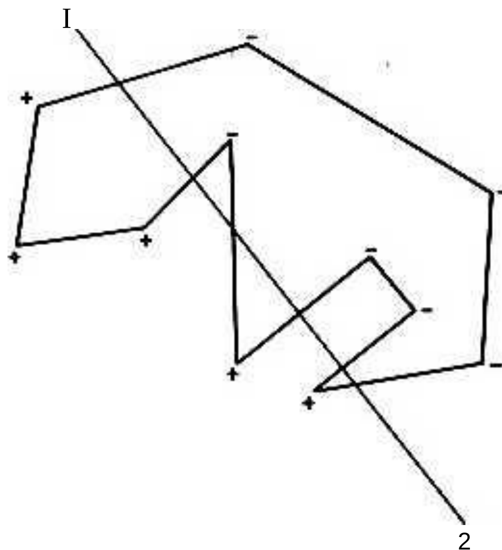


Figure 15.3 **Illustration of a polygon clipped by a line. Each vertex is marked with the sign of U_t computed there.**

15.5 INTERSECTIA A DOUA POLIGONI

Găsirea intersecției a două poligoane prezintă un mare interes în soluționarea problemelor de vizibilitate și reprezintă formularea cea mai generală a problemei de tăiere. Problema poate fi rezolvată prin aplicații repetate ale algoritmului 15.3. Fie cele două poligoane fl| si II? și să presupunem, de asemenea, că n $_2$ este convex. Atunci este posibilă următoarea soluție. Dacă q $_b$ q $_2$, ' ' ' q_n sunt laturile lui n $_2$, iar P_t, i ■ 1. 2. ■ ■ ■ m sunt vârfurile lui n $_h$ care pot fi sau nu convexe, atunci putem aplica algoritmul 15.3 pentru fiecare dintre laturile <7, pentru poligonul II|. Rezultatul tăierii II] de către gi va fi orice număr de poligoane, de la zero la unu peste numărul de vârfuri concave ale lui II,. Din cauza posibilei creșteri a numărului de poligoane, este necesară o contabilitate atentă.!

Cel mai bine este să folosiți o listă legată pentru a descrie n $_{lt}$ și orice alte poligoane care pot fi găsite. În special, cu fiecare vârf asociem un pointer la următorul vârf atunci când poligonul este parcurs în a

t Cititorul atent poate obiecta la această metodă subliniind că algoritmul 15.3 decupează un poligon printr-o linie mai degrabă decât un segment de linie. Totuși, presupunerea noastră că n $_2$ este convex face o astfel de substituție legală.

mod în sensul acelor de ceasornic. Vom folosi simbolul $NXT(P_t)$ pentru a indica acel pointer. Initial avem $NXT(P_t) = P_{I+i}$ pentru $i < m$ si $NXT(P_m) = P$^ Continuam acum la modificarea algoritmului 15.3. În loc să desenăm laturile, vom actualiza pur și simplu indicatoarele respective. În special, la pasul 6 adăugăm în listă punctul de intersecție și setăm $NXT(P_m +/) = P_t$, unde j este un număr care indică poziția noului punct la sfârșitul unei liste. În pasul 7 setăm NXT^Pt-^ = $P_{m+}j$ și $NXT(P_{m+J})$ - 0. deoarece nu știm încă

următorul punct. Dacă un ver tex nu este vizibil, setăm și valoarea indicatorului său la zero. Figura 15.4 ilustrează o parte a acestui proces utilizând săgeți grele pentru a indica efectul pointerelor. Pentru a finaliza procesul, trebuie să stabilim indicatori între punctele de intersecție. În acest scop sortăm punctele P_{m+j} în ordinea creșterii y, sau crescând x dacă linia de tăiere este orizontală. Apoi le luăm pe perechi și schimbăm oricare dintre indicatori este zero la adresa celuilalt element al perechii. Astfel, în cazul figurii 15.4 adăugăm următoarele două legături:

$$NXT(P_{m}\hat{}) - P_{n+A}\ NXT\{P_{m}\hat{}) - P_{m}\hat{}$$

Acum este un proces simplu să parcurgeți lista de vârfuri și să găsiți noile poligoane. De fapt, acest lucru nu trebuie făcut până la sfârșit, deoarece algoritmului 15.3 nu prea îi pasă câte poligoane separate există. Probabil cea mai bună modalitate de a parcurge lista de vârfuri este să folosiți un al doilea set de pointeri care să conțină adresa din lista unui singur punct din fiecare poligon. Acest set poate fi scanat secvenţial, iar următorii indicatori pot fi utilizaţi pentru a parcurge conturul fiecărui poligon.

Procesul de mai sus găsește intersecția celor două poligoane prin găsirea intersecției lui 11, cu semiplanurile ty definite de laturile lui n_2. (Dreapta care conține latura q_t împarte planul în două părți. ty este partea care conține puncte din interiorul poligonului lângă *q,.)* Dacă II_2 nu este convex, atunci rezultatul aplicării acestei metode este intersecția lui ty, nu cu n_2, ci cu intersecția semiplanurilor: /V — Qty. ^ se numește *nucleul* sau *nucleul* lui $n_?$. Este un poligon convex care are proprietatea că din oricare dintre punctele sale se poate trage o linie la oricare dintre vârfurile lui n_2 și are linia în întregime în n_2. (Desigur, nucleul poate fi un set gol.) O discuție despre nucleu și proprietățile sale depășește scopul acestui text (vezi Notele bibliografice).

Pe de altă parte, algoritmul 15.3 poate fi folosit pentru a descompune un poligon neconvex în componente convexe. În special, fie q_t și *q^i* laturile unuia dintre unghiurile concave ale unui poligon n. Tăierea n_2 mai întâi cu q_f și apoi cu q_{J+l} produce două poligoane: II Qty și

n Q ty+p Fiecare dintre ele are cel puțin un unghi concav mai puțin decât II. Dacă repetăm acest proces pentru toate laturile cu unghiuri concave, obținem o colecție de poligoane a căror unire este II. Aceste poligoane pot fi tăiate cu laturile unghiurilor lor concave și așa mai departe, până când toate poligoanele rezultate sunt convexe (vezi Problemele 15.6 și 15.7).

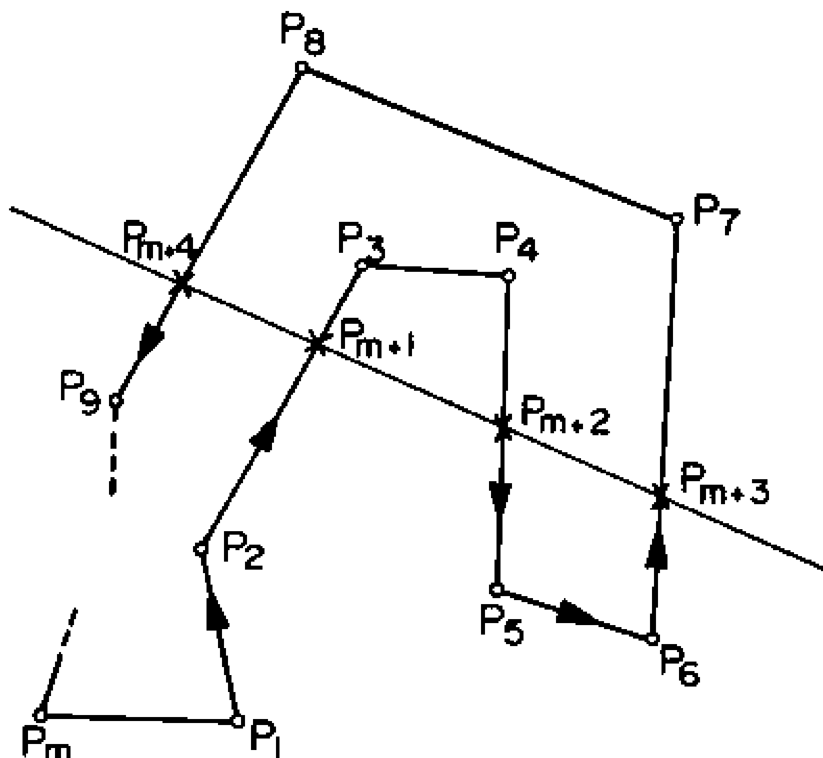


Figura 15.4 **Ilustrarea setării indicatorului pentru tăierea unui poligon cu o linie**

În loc să încercăm o descompunere convexă a lui n 2, putem încerca să modificăm aplicarea algoritmului 15.3 după cum urmează. Fiecare dintre laturile lui II 2 este folosită pentru a tăia poligonul II b original , mai degrabă decât poligoanele rezultate din intersecțiile anterioare. La sfârșit, punctele de intersecție de-a lungul laturilor lui II 2 sunt sortate pentru a forma intersecția poligoanelor. O tratare completă a acestui subiect este în afara domeniului nostru de aplicare, nu numai datorită complexității sale, ci și pentru că în majoritatea aplicațiilor grafice poligonul n 2 fie corespunde ferestrei de afișare (un dreptunghi obișnuit), fie este proiecția unui triunghi sau cvadangle (vezi Capitolul 17).

15.6 INTERSECȚIE EFICIENTĂ DE POLIGON

Găsirea intersecției a două poligoane prin aplicații repetate ale algoritmului 15.3 așa cum este descris în secțiunea anterioară nu este întotdeauna eficientă. De exemplu, nu transmitem nicio informație despre locația vârfurilor lui II, de la o aplicație a

algoritmului la alta. Există mai multe modalități de îmbunătățire a eficienței procesului și vom prezenta câteva dintre ele aici. Reamintim că *n* și *m* au fost folosite pentru a desemna numărul de vârfuri (sau laturi) ale lui II $_2$ și, respectiv, 11. În cel mai rău caz, pașii 3-8 ai algoritmului 15.3 vor fi executați de *câteva* ori. Efortul de găsire a intersecției va fi cel puțin +n, deoarece trebuie verificate toate vârfurile cel puțin o dată. Diferența dintre *mn* și *m + n* poate fi substanțială și acest lucru invită la o investigație pentru proceduri eficiente.

Cu toate acestea, ar trebui să remarcăm că, dacă m este un număr mic (de exemplu, trei sau patru), așa cum este adesea cazul în aplicațiile grafice, este posibil ca o creștere a eficienței să nu fie posibilă. Orice este salvat în reducerea numărului de vizite ale unui vârf se poate pierde în efortul sporit de păstrare a contabilității. Astfel, discuția din restul acestei secțiuni este relevantă pentru problema găsirii intersecției a două poligoane numai atunci când ambele au un număr mare de vârfuri. Ceea ce este important în grafică sunt algoritmii pentru găsirea eficientă a intersecțiilor perechi ale unui număr mare de poligoane. Acesta este cazul soluțiilor problemei de vizibilitate și cu verificarea layout-urilor circuitelor, mai ales când se utilizează integrarea la scară largă. Principiile folosite pentru algoritmii eficienți de intersecție a poligoanelor sunt de asemenea aplicabile acolo. Din acest motiv, descriem câteva dintre metodele majore utilizate.

O posibilă accelerare în găsirea intersecțiilor este utilizarea unor forme mai simple decât poligoanele originale. Fie *My* și *M* $_2$ *dreptunghiurile regulate* circumscrise ale poligoanelor II și n $_2$. (Acestea pot fi găsite cu ușurință din valorile extreme ale coordonatelor vârfurilor poligoanelor.) În loc să rezolvăm problema intersecției pentru poligoane se poate rezolva în schimb pentru *M* și *M* $_2$*,* o sarcină mult mai ușoară. Dacă dreptunghiurile nu se intersectează, atunci știm că nici poligoanele nu se intersectează. Dacă unul dintre dreptunghiuri este conținut în altul, să spunem că *M* $_2$ este în *M* ,. atunci putem lua în considerare intersecția dintre *M* $_2$ și JI, și apoi intersecția ultimului poligon și n $_2$ (Figura 15.5a). Dacă *My* și *M* $_2$ au o intersecție nevidă A/ $_{12}$, putem lua în considerare intersecția fiecărui poligon cu recunglc *M y* $_2$*,* apoi definim două poligoane noi și le aplicăm întreaga procedură (Figura 15.5b). Această abordare are avantajul că elimină rapid din considerare vârfurile poligoanelor care sunt departe unul de celălalt.

Alte abordări depind de presortarea vârfurilor și majoritatea economiilor apar atunci când cel puțin unul dintre poligoane rămâne fix.

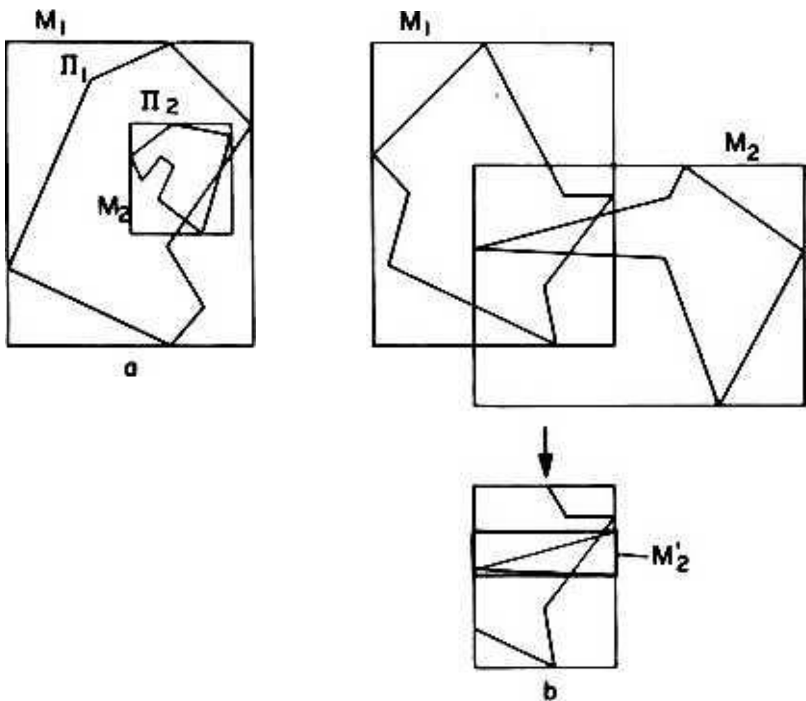


Figure 15.5 **(a) Definition of new polygons for checking the intersection of II| and 11,. (b) Use of recursive subdivision for clipping.**

Prima metodă, propusă în [15.DL], este destul de generală și poate fi aplicată la diverse probleme geometrice. Fie *L\ t L 2. "* • să fie o mulţime de drepte pe plan şi P b *P 2 , * * ' ?** 'fii puncte distincte ale intersecțiilor lor perechi. În mod clar, *h* < A(A:+l)/2. Fie x,.^, coordonatele lui *P t .* Putem presupune fără pierderi de generalitate că punctele de intersecție au fost etichetate astfel încât să aibă x 4 < x <+) . Liniile verticale x = x (împart planul în *h* + 1 plăci cu proprietatea că nu există intersecții de linii în cadrul fiecărei plăci. Acum este posibil să ordonați segmentele de linie în interiorul fiecărei plăci în funcție, de exemplu, coordonatele j a punctului lor mijlociu. Această ordonare este bine definită deoarece nu există intersecții. Costul ordonării *K* obiectelor este de ordinul Klog ^ în acest caz va fi de ordinul K plus A log.

Să presupunem că acum avem un punct dat (x,>') și dorim să aflăm dacă acesta aparține vreuneia dintre cele *k* linii, sau se află în interiorul unei regiuni definite de acele linii. Putem rezolva această problemă găsind mai întâi placa căreia îi aparține punctul pe baza coordonatei sale x, apoi plasând-o în raport cu liniile pe baza coordonatei sale *y* .

Ambele plasări sunt echivalente cu inserarea într-o listă ordonată, problemă care necesită timp proporțional cu logaritmul mărimii listei.t În cazul nostru, aceasta este egală cu $\log_2 A$ plus $\log_2 L$ Deoarece $\log_2 A < 2\log_{2A}$ putem spune că efortul este proporțional cu $\log_2 L$ (Algoritmul naiv necesită efort de ordinul lui *k).*

Teorema 15.1: Este posibil să se verifice poziția unui punct față de *k* drepte în timp de ordinul $\log_2 t$, cu condiția ca liniile să fi fost presortate. Efortul de presortare este de ordinul $t'\log_2 L$ [15.DL] □

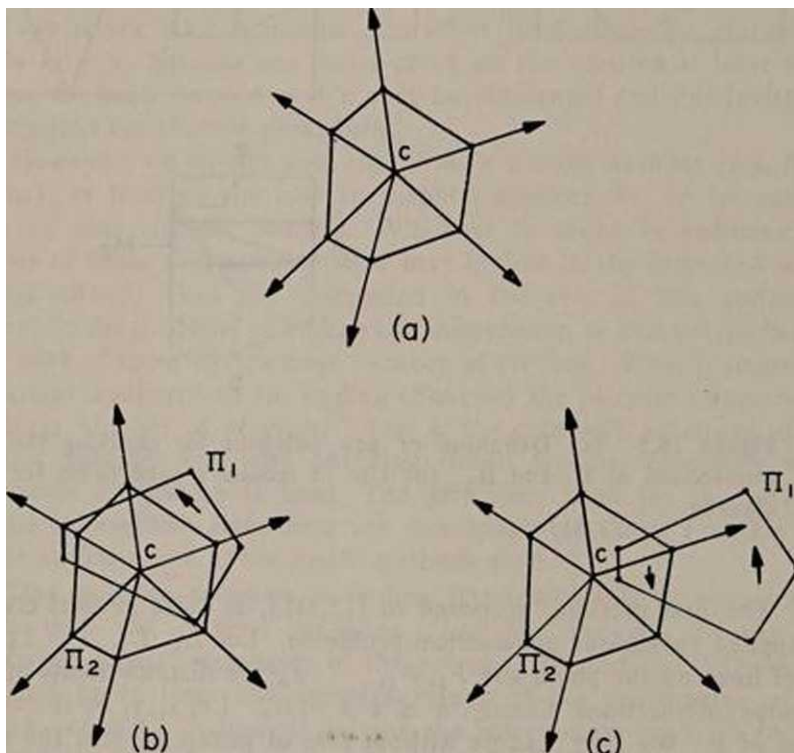


Figura 15.6 (a) Definiția sectoarelor pentru găsirea intersecției a două poligoane convexe, (b) Determinarea intersecției când C se află în interiorul ambelor poligoane. Pentru ambele poligoane poate fi stabilită o direcție comună de deplasare secvențială , (c) Găsirea intersecției când C este în afara II|. Două direcții de traversare secvențială pentru II I trebuie utilizate, dar doar unele dintre sectoarele II_2 trebuie vizitate.

t Aceasta presupune că lista este menținută într-o structură de date adecvată, cum ar fi un arbore echilibrat. Vezi [6.AHU). Capitolul 3 pentru mai multe despre acest subiect.

O altă metodă, propusă în [15.SH], face o presortare similară, dar în coordonate polare. Are restricția severă că ambele poligoane trebuie să fie convexe. Se procedează prin selectarea unui punct *C* din interiorul unuia dintre poligoane, să spunem n_2, și găsește semiliniile $L_h L_2, ''' L_m$ care unesc *C* cu *m* vârfurile lui II_2. Efortul pentru găsirea punctului și a dreptelor este proporțional cu m. Dacă liniile sunt direcționate de la centru către vârf, astfel încât punctele din dreapta să dea un semn pozitiv în ecuația (15.1), atunci este simplu să verificați dacă un punct aparține unui sector. Acesta va fi în sectorul liniilor $L_($ și $£_{/+l}$ dacă este pozitiv față de primul și negativ față de cel de-al doilea (Figura 15.6a). Sectoarele pot fi ordonate, de exemplu, în sensul acelor de ceasornic, astfel încât să se găsească sectorul căruia îi aparține un punct în timp proporțional cu $\log_2 m$. Odată ce sectorul pentru un vârf al celui de-al doilea sector a fost găsit, să zicem poligonul nj rămâne pentru vârful nj. găsite luând în considerare secvențial secvenții pentru a vedea dacă acestea conțin vreun vârf. Dacă 11) conține centrul (Figura 15.6b) sectoarele și vârfurile lui II l. sunt parcurse împreună fără a fi nevoie vreodată să dea înapoi. Dacă centrul este în afara II] (Figura 15.6c), atunci se poate găsi un nou sector care conține II] și centrat pe *C.* Vârfurile lui II] se găsesc acum în două grupuri și pot fi examinate secvenţial pentru a-și găsi sectoarele. În ambele cazuri, procesul necesită timp proporțional cu *n,* unde *n* este numărul de vârfuri ale lui II|. După ce acest proces este terminat, se examinează poziția vârfurilor din fiecare sector față de latura II_2 de acolo.

15.7 NOTE BIBLIOGRAFICE

[I.NS] descrie un algoritm de tăiere a liniilor cu privire la un dreptunghi obișnuit care utilizează subdiviziuni recursive. (15.SUH] are o tratare detaliată a algoritmului 15.3 și a altor probleme înrudite. [15.WE] discută problema generală a operațiilor între poligoane: împărțirea unui poligon în părți, unirea a două poligoane care se intersectează etc. În mod clar, astfel de probleme pot fi rezolvate cu ușurință prin variații ale algoritmilor de tăiere, dar se poate evita o strategie adecvată de calcul a mai multor părți a poligonului.

Utilizarea descompunerii poligoanelor în componente convexe pentru recunoașterea modelelor este descrisă în [3.PA].

Teorema 15.1 este o parafrază a lui Dobkin și Lipton (15.DL). Al doilea algoritm eficient de intersecție se datorează lui Shamos [15.SH]. Chiar dacă acest algoritm și altele asemănătoare sunt asimptotic (deoarece numărul de vârfuri de poligon tinde spre infinit) mai rapid decât cei descriși în acest text, aplicabilitatea lor directă în grafică și recunoașterea modelelor este îndoielnică din motivele discutate în

Secțiunea 15.6. De asemenea, poligoane cu multe vârfuri (depășind, să zicem, zece) care sunt întâlnite în grafice sunt probabil să nu fie convexe. Prin urmare, algoritmii eficienți asimptotic pentru problema intersecției sunt de interes acolo numai dacă sunt aplicabili la poligoane neconvexe. Pe de altă parte, principiul presortării și utilizarea dreptunghiurilor circumscrise pot fi utile atunci când se lucrează cu un număr foarte mare de poligoane. Acolo putem folosi o metodă simplă, cum ar fi algoritmul 15.3, pentru a găsi intersecția reală, dar aplicăm un algoritm eficient asimptotic pentru restrângerea numărului de poligoane care trebuie examinate . Consultați (I5.CD) pentru metodele de detectare a intersecțiilor, mai degrabă decât pentru calcularea acestora.

15.8 LITERATURA RELEVANTA

(15.CD] Cbazclle, B. și Dobkin, DP „Detection is Easier than Compu tation," *Proc. Twelfth Annual ACM Symposium on Theory of Computing ,* **Los Angeles, California, 28-30 aprilie 1980, pp. 146-153.**

(IS.DL] Dobkin. D. şi Lipton, RJ "Multidimensional Searching Prob lems", *SIAM J. Computing,* **5 (1976), pp. 181-186.**

[15SH| Shamos, M. L „Complexitatea geometrică" *Proc. Al șaptelea simpozion anual ACM despre teoria calculului,* **1975, pp. 224-233.**

[I5SUH1 Sutherland, IE și Hodgman, GW „Reentrant Polygon Clip ping", *CACM,* **17 (1974). pp. 32-42.**

[)5.WEj Weiler, K. „Polygon Comparison using a Graph Representation," *Computer Graphics.* **14 (iulie 1980), p. 10-18. (Proceedings of SIG GRAPH 80, publicat de ACM.)**

15.9 PROBLEME

15.1. Investigați algoritmul de tăiere pentru cazul special când linia *L* trece prin unul dintre vârfurile poligonului. Sunt necesare modificări?

15.2. Ce modificări ar trebui făcute în algoritmul 15.1 dacă segmentul de linie este înlocuit cu o linie (infinită) dată de coeficienții săi?

15.3. Scrieți un program care implementează algoritmul 15.2.

15.4. Modificați algoritmul 15.3 astfel încât să deseneze un vector între intersecțiile succesive ale poligonului cu linia. Apoi figura rezultată va fi închisă și poate fi completată utilizând unul dintre algoritmii din Capitolul 8. *(Sugestii:* este necesar să păstrăm un steag care să indice pe ce parte a poligonului ne aflăm. Când steagul este schimbat, punctul de intersecție este salvat. Data viitoare când steagul este schimbat, un vector este desenat între punctul salvat și noua intersecție trebuie să fie tratat corespunzător.

situații precum un vârf de poligon situat pe linie și inițializarea corectă. Este utilă stabilirea unei ordini de parcurgere a vârfurilor în sensul acelor de ceasornic.)

15.5. Proiectați și implementați un algoritm pentru găsirea intersecției și unirii a două poligoane. Rezultatele trebuie exprimate ca un set de poligoane. (Setul poate fi gol pentru intersecție .)

!5.6. Proiectați și implementați un algoritm pentru descompunerea unui poligon (neconvex) în submulțimi convexe. Urmați sugestia dată în Secțiunea 15.5 și demonstrați că unirea poligoanelor rezultate este într-adevăr poligonul original. Oferiți o estimare a complexității de calcul a algoritmului.

15.7. Proiectați și implementați un algoritm pentru descompunerea unui poligon (neconvex) în submulțimi convexe cu constrângerea adăugată că componentele convexe nu se intersectează. Comparați complexitatea comparațională a acestui algoritm și cea a problemei anterioare.

15.8. Numărați exact câte ori trebuie evaluate expresiile ecuației (15.1) sau ecuațiilor (15.2) pentru a găsi intersecția a două pentagoane, mai întâi folosind metoda din secțiunea 15.5 și apoi una dintre metodele din secțiunea 15.6.

Capitolul 16

MATEMATICA GRAFICII TRIDIMENSIONALE

16.1 INTRODUCERE

Grafica pe computer oferă un exemplu interesant de renaștere a unor metodologii și ramuri ale matematicii aproape uitate. Redarea scenelor tridimensionale pe o suprafață plană a preocupat pictorii de multe secole și studiul proiecțiilor în perspectivă au ocupat o poziție proeminentă în școlile de artă. Pe la începutul secolului al XIX-lea, studiul *geometriei descriptive* a devenit un subiect central în inginerie și s-a ocupat de probleme precum găsirea intersecției cilindrilor prin mijloace grafice etc. Invenția fotografiei a redus semnificația artei realiste, iar apariția artei abstracte a plasat studiile proiecțiilor în fundal. În mod similar, geometria descriptivă și-a pierdut proeminența în inginerie și, în 1960, puțini studenți de inginerie din Statele Unite studiau materia , f

Grafică pe computer care creează reprezentări bidimensionale

t Într-un fel. Am avut norocul să-mi termin studiile de licență într-o țară în care programa s-a schimbat puțin cu var. Geometria descriptivă a fost accentuată la Universitatea Tehnică din Atena până în anii cincizeci.

pentru scenele tridimensionale a reînviat interesul pentru aceste subiecte. Instrumentele matematice majore necesare pentru crearea de afișaje care apar tridimensionale includ metode pentru rezolvarea problemelor geometrice (discutate în Secțiunea 16.2), transformările poziționale (Secțiunea 16.3) și proiecțiile (Secțiunile 16.4 și 16.5).

Cunoștințele matematice sunt doar un instrument necesar și în niciun caz suficient pentru a proiecta afișaje bune. O bună judecată inginerească , precum și o anumită abilitate în artele vizuale sunt alte premise esențiale . În ultimii zece sau cincisprezece ani a existat un interes substanțial pentru ceea ce se numește arta computerizată. În cea mai mare parte, astfel de creații erau bidimensionale. Poate cea mai mare provocare tehnică este utilizarea computerului pentru a crea scene care par tridimensionale. Cantitatea mare de calcul necesară este un impediment pentru astfel de eforturi; mai mult accent pe studiul instrumentelor matematice utilizate poate duce la implementări mai eficiente.

16.2 COORDONATE OMogene

Analiza secțiunii 14.3 poate fi extinsă la trei dimensiuni. Un punct *P* poate fi descris prin coordonatele omogene *x,y,z,w* iar un plan /I prin coeficienții *a, b, c, d,* astfel încât ecuația planului să fie

În trei dimensiuni, dualitatea este între punct și plan. Astfel, trei puncte definesc un plan printr-o ecuație de formă

ax 4- *by* 4- *cz* + *dw* = 0 (16.1)

det |X *1 *2 *3 | y 3'1 ^2 ^3 | z *1 z 2 *3 | w ^1 *2 *3| = 0 (16,2)

iar trei planuri se intersectează la o punct cu coordonate care sunt minori de următorul determinant.

det |o «1 *2 *3 | *b* *1 ^2 63 | *c* ®1 ^2 ®3 | *w* w! ^2 ^3| -0 • (16,3)

Aceste definiții eșuează în cazuri singulare: dacă cele trei puncte se află pe o dreaptă, dacă cele trei plane au o dreaptă comună sau dacă cele două puncte sau planuri care definesc o dreaptă coincid. În ceea ce privește ecuația (16.2), condițiile de singularitate se manifestă prin faptul că una dintre coloanele 2 la 4 este o combinație liniară a celorlalte două și, prin urmare, determinantul este zero pentru toate valorile lui *(x,y,z,w).*

Liniile necesită acum o pereche de ecuații, fie ca intersecții a două plane, fie ca unind două puncte. Pentru a exprima ecuația unei drepte care unește două puncte în coordonate omogene, observăm că o astfel de dreaptă ar trebui să aparțină tuturor planurilor definite de (X|j'|,z I ,W|), (x 2 ,.y 2 ,z 2 ,w 2) și orice al treilea punct. Prin urmare, ecuația (16.2) ar trebui să fie o identitate în ceea ce privește valorile elementelor din a patra coloană. Aceasta înseamnă, la rândul său, că determinanții minori cu privire la elementele acelei coloane ar trebui să fie zero. Cu toate acestea, cele patru ecuații rezultate nu sunt toate independente. Într-adevăr, luând în considerare minorul față de x 3 concluzionăm că rândul (M»,W|,H' 2) este liniar dependent de rândurile O'J» 1^2) și (^^^1^2) > n ordinul ca deter minantul să fie zero. În mod similar, constatăm că (B',w» 1 ,w 2) depinde liniar de (x,x lt x 2) și (z,Z|,z 2). Aceste două dependențe liniare implică dependența liniară a lui (x,x t ,x 2), O'^ 1^2)» și (z,z b z 2). Prin urmare, minorul față de w 3 este zero, dacă minorii față de x 3 și y 3 sunt zero. Demonstrația poate fi repetată pentru minor față de z 3, astfel încât să rămânem cu două ecuații independente:

Dacă setăm w — wj = w 2 = 1 în ecuațiile de mai sus, obținem expresiile obișnuite pentru o dreaptă care unește două puncte:

det |X x, x2 | y ^1 >'2 | **W Wi w2**| = det |X X| x2 | Z Zi z2 | **w W] w2**| = 0 (16.4)

XX| **y-y!** **Z - Z;**

2"| ^|"^! z 2" z |

16.2.1 Poziția unui punct față de un plan (16.5)

Pentru a verifica poziția unui punct față de un plan trebuie să evaluăm semnul determinantului ecuației (16.1) pentru coordonatele punctului. Problema în trei dimensiuni este mai complicată decât în două și avem nevoie mai întâi de anumite preliminarii.

Definiția 16.1: Fie v b V 2 » și v 3 trei vectori, toți pornind dintr-un punct comun și nu se află pe același plan. Vom spune că acești vectori formează un *sistem dreptaci* dacă au următoarea proprietate. Fie *P* planul definit de vj și v 2 , iar *H* semispațiul definit de *P* unde se află v 3 . Apoi, pentru un observator situat în *H,* și situat în punctul comun al celor trei vectori, vectorul Vj poate fi adus la v 2 printr-o rotație în sens invers acelor de ceasornic. □

Alegerea obișnuită a axelor xy-2 are ca rezultat un astfel de sistem. O ilustrare comună a unui sistem cu mâna dreaptă este oferită de degetul mare, arătător și mijlociu al unei mâini drepte, dacă sunt atribuite v„ §2 . și, respectiv, v 3. Când degetele sunt ținute în așa fel încât să fie reciproc perpendiculare, ele formează un sistem cu mâna dreaptă. O modalitate analitică simplă de verificare a acestei proprietăți este oferită de următoarea lemă.

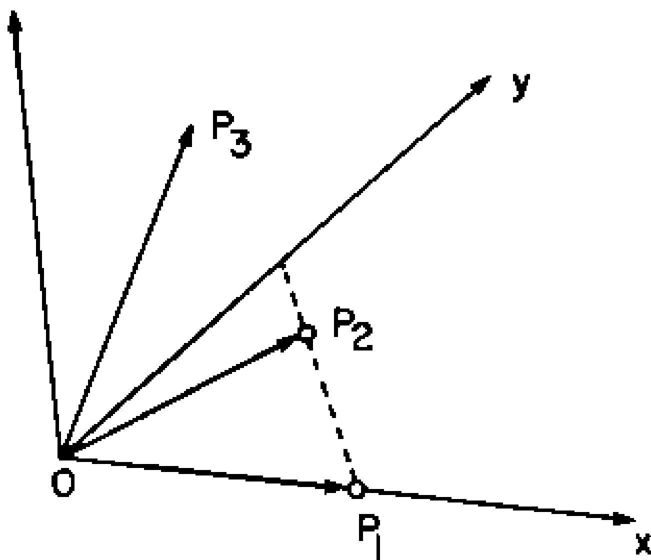


Figure 16.1 Arrangement of vector* forming a right system

Lema 16.1: Valoarea *d* a determinantului

*i *2 *1

d - det yi y2 yj (16.6)

§1 *2 Xj

este pozitivă dacă cele trei puncte formează un *sistem dreptaci* în raport cu originea. □

Dovada: Fără nicio pierdere de generalitate putem lua axa x de-a lungul dreptei care unește originea cu primul punct, axa y pe planul definit de primele două puncte și formând un unghi mai mic de 90 de grade cu linia care unește originea cu al doilea punct și axa z normală la acel plan și formând un sistem de două axe drepte (vezi figura16). Atunci ecuația (16.6) devine

*i xi xj

d - **det** 0 ^i Pl “X^^ (16.60

0 0 »3

Datorită selecției noastre a axelor, toate cele trei coordonate din partea dreaptă a ecuației de mai sus sunt pozitive și, prin urmare, același lucru este valabil și pentru produsul lor. □

Se poate arăta (vezi (16.SA], p.21) că *d* este egal de șase ori volumul unei piramide a cărei bază este triunghiul format din cele trei puncte și al cărei vârf este originea.

Următorul este o contrapartidă a Ipotezei 14.1.

Ipoteza 16.1: Dacă un plan este specificat de trei dintre punctele sale, atunci aceste puncte sunt ordonate astfel încât să formeze un sistem de dreapta în raport cu originea. □

Propoziția 16.1: Fie A, *Y, Z* și IF coordonatele unui punct *P* și *(XtMa^Wj), l* — 1, 2, 3 trei puncte care definesc un plan. Dacă *W* și toate w (s sunt pozitive, atunci *P* se află de aceeași parte a planului cu originea dacă valoarea determinantului dat de ecuația (16.7) este pozitivă.

X| x 2 X 3 P2 Pl *2 *3 *2 *3

Pl □ (16.7)

»1

*1

Demonstrație: Partea dreaptă a ecuației (16.7) este o funcție liniară a lui x, *y, z* și w și, prin urmare, semnul său va fi fixat de fiecare parte a planului definit de D — 0. Înlocuind *X — Y — Z — Q* în această ecuație găsim

D -Wd (16,8)

unde *d* este determinantul dat de ecuația (16.6). Datorită ipotezei 16.1, Lemei 16.1 și ipotezei despre semnul lui *W,* ambii factori ai produsului sunt pozitivi. □

Rezultatul de mai sus poate fi extins astfel încât să poată fi utilizat pentru a defini poziția unui punct în raport cu suprafețele neplane. Într-adevăr, aproape orice suprafață poate fi subdivizată în triunghiuri ale căror vârfuri pot fi marcate într-un mod consistent, astfel încât să formeze un sistem dreptaci în raport cu originea. Singurele excepții sunt suprafețele precum

*Banda Moebius** care nu sunt obișnuite în aplicațiile grafice (a se vedea [16.PF], p.214 pentru mai multe despre orientarea suprafeței).

16.2.2 Intersecția triunghiurilor

Introducem următoarea notație pentru un determinant:

$$Silk^* \text{ “ } \begin{vmatrix} x_i & x_J & *k & X* \\ >'i & y_j & y* & y_„ \\ z_i & z/ & *k & z* \\ \% & w_i & *k & \end{vmatrix} \quad (16.9)$$

Propoziţia 16.2: Se consideră două triunghiuri date de trei puncte fiecare: Pi, P_2, $P_}$ primul triunghi, iar P_{Ai} $P\$*$ P_6 al doilea. Fie x_t * $y_{,,}$ z_{ly} și w_t coordonatele omogene ale *lui Pi* (l<r<6). Atunci cele două triunghiuri se intersectează dacă cele trei mărimi Suss. 2456 USD. ^{și} d ^M$6 d° ^{nu} toate au acelaşi semn şi aceeaşi condiţie este valabilă pentru cele trei mărimi S_4 nj, S_{J123} şi S_6 | $_{2J}$. □

Dovada: O consecință directă a Propoziției 16.1. □

Acest rezultat este o generalizare a discuției din Secțiunea 14.4.2 de la două la trei dimensiuni.

16.3 TRANSFORMĂRI TRIDIMENSIONALE

Translația și transformările de scalare în trei dimensiuni pot fi exprimate în același mod ca și în două dimensiuni, dar pentru rotații , trebuie să alegem acum o axă. Formulele de transformare a coordonatelor după o rotație în jurul unei axe arbitrare sunt destul de greoaie, dar este posibil să le oferim în formă concisă cu dovezi destul de simple dacă introducem anumite concepte de algebră vectorială.

16.3.1 Preliminari matematice

Am presupus de-a lungul acestui text că cititorii sunt familiarizați cu algebra liniară elementară și am folosit deja conceptele de produs scalar și produs al unei matrice și al vectorilor în secțiunile anterioare (Secțiunea 2.2, Secțiunea 10.3 etc.). Prezentăm aici câteva operații vectoriale suplimentare, care, deși sunt utilizate pe scară largă în mecanică și teoria electromagnetică, ar putea să nu fie foarte familiare studenților în informatică sau procesarea semnalului.

Definiția 16.2: Produsul încrucișat al a doi vectori tridimensionali M și v, notați MXV, este definit ca un vector cu componente

$$^u y\ ^v t\ ^- {}^u x\ ^v y*\ ^u x\ ^v x\ ''\ \hat{}\ x\ v\ f\ \hat{}x\ ^{vt}\sim\ ^u t\ ^v x\ \bullet \quad (16.10)$$

Următoarea propoziție rezumă anumite proprietăți ale produsului încrucișat. Dovezile sunt simple.

Propoziția 16.3: (a) **Dacă** M şi v sunt coliniare, adică $u = cv$ pentru o constantă c, atunci KXV =0.

(b) Dacă M și v nu sunt coliniare, atunci MXV este normal cu planul definit de ele.

(c) MXF - -VXM.

(d) Mărimea produsului încrucișat este egală cu 1« Hv Isind, unde 0 este unghiul dintre cei doi vectori. □

Definiția 16.3: Produsul încrucișat al unui vector **v** și a unei matrice >**4** este o matrice ale cărei coloane sunt egale cu produsul încrucișat al coloanelor lui A cu v. □

Observăm, de asemenea, că produsul unui vector A -dimensional v cu transpunerea u' a altui vector n-dimensional este o matrice IF ale cărei elemente sunt date de ecuație

$$^tj „ v_t «j \tag{16.11}$$

Deoarece vectorul coloană este o matrice nXl și vectorul rând o matrice IX«, produsul de mai sus va fi o matrice »Xn. (Dacă ordinea operației este inversată avem o matrice 1X1, adică o scalară.) Această operație vectorială are un nume special:

Definiția 16.4: O *diada* este produsul unui vector coloană u cu un vector rând v': MV'. Reprezentarea *diadică* a unei matrice /I este forma

$$A '' a^a\ + a^a^ + \blacksquare \cdot \cdot + a_k a_k'$$

pentru unii vectori O|, «2, *••□*. □

Vom folosi conceptul de diade pentru a deriva expresia transformării coordonatelor cauzate de o rotație în jurul unei axe prin origine. O discuție despre reprezentările diadice este în afara sferei textului și ne limităm să dăm doar un exemplu. Rețineți că nu toate matricele au reprezentări diadice.

Exemplul 16.1: Matricea

$$^ - \begin{matrix} 2 & 0 \\ 1 & 3 \end{matrix}$$

are reprezentarea diadică

'Hoh ² - ²) ⁴ -^ ⁰ ')

Pe de altă parte, nu există nicio modalitate de a exprima matricea

$$'' \begin{matrix} 2 & 1 \\ 0 & 2 \end{matrix}$$

într-o formă diadică. □

16.3.2 Zona de rotație și o axă prin Origia

Vom studia această problemă mai întâi în coordonatele x, y, *z .* Fie ca o axă prin origine să fie definită de unghiurile 4i« 0 $_{3l}$ și 0, pe care le formează cu cele trei axe de coordonate x, y și *z.* Rețineți că aceste unghiuri nu sunt independente, deoarece următoarea ecuație este valabilă pentru cosinusurile lor:

$$\text{cos'^i + cot}^2 4_2 + \cos^2 \hat{}_3 \text{"I} \qquad (16.12)$$

Pentru axa dată definim un vector a care este paralel cu axa și care are lungimea unitară. Atunci a_t— cos^, /■ 1,2,3.

Teorema 16.1: Matricea transformării de coordonate impusă de rotația cu un unghi 9 în jurul unei axe prin originea paralelă cu vectorul unității de lungime a este dată de

rt(0) - a«' + coi0(/ -oa) + sinö(/Xa) (16.13)

unde / este matricea de identitate 3X3. □

Dovada: Fie 6 un vector de la origine care va fi rotit în jurul axei. Este întotdeauna posibil să se descompună *b* în două componente, una paralelă cu axa și alta, *d,* normală cu aceasta, astfel încât 6 poate fi scris ca sumă

b-Ka+d (16,14)

pentru o constantă X. Calculăm produsul *Rb* utilizând expresiile ecuațiilor 16.13 și 16.14.

Rb = Xoa'a + Xcos0(« -ooa) + XsinG(/Xa)a

+ *aa'd* + cos0(rf - *aa'd)* + sind(/Xa)^ (16.15)

Toate operațiile asupra vectorilor sunt asociative și produsul unei diade *ef* cu un vector *g* este egal cu e ori produsul scalar *al lui f'g.* Dacă aplicăm acest termen de rearanjare tuturor termenilor din ecuația (16.15) putem simplifica acea ecuație în mod semnificativ. Deoarece a s-a presupus a fi de unitate de lungime, observăm că *a'a* - I, iar pentru că *d* a fost ales ca fiind ortogonal cu a, avem *a'd* -0. De asemenea (/Xu)« trebuie să fie zero deoarece toate coloanele lui *l*a* sunt ortogonale cu a. Apoi cele de mai sus

equauo^comes

Rb " Xa* + cos0J + sin0(/Xa)</ (16.16)

Pentru a finaliza demonstrația teoremei trebuie doar să arătăm că ultimii doi termeni reprezintă o rotație a lui *d* cu un unghi 0 pe planul perpendicular pe axă. În primul rând, observăm că *(iXa)d = dxa.* Putem arăta că formal calculând în mod explicit matricea *ixa*

0 *3-*2

iXa - —aj 0 a I

aj ~~ ^{a} I 0

iar apoi greblarea produsului cu *d.* Prin urmare, ultimii doi termeni pot fi scriși ca

cotM + sin0<fXa (16.17)

Vectorul *dXa* este perpendicular atât pe *d* cât și *pe a* și are lungimea egală cu lungimea lui *d* din cauza Propoziției 16.3d. Dacă introducem un sistem de coordonate local pe planul perpendicular pe axă, atunci putem alege o axă paralelă cu *d* și cealaltă paralelă cu *dXa* , astfel încât ecuația (16.17) să fie într-adevăr o rotație cu un unghi *6* pe acel plan. Aceasta completează dovada. □

Ecuația (16.16) poate fi extinsă astfel încât matricea de rotație să fie exprimată în termeni de unghiuri 0 și ^. De asemenea, putem adăuga un rând și o coloană suplimentară, astfel încât să poată fi aplicate la coordonate omogene. Atunci o rotație cu un unghi 0 este dată de o matrice 4X4 ale cărei elemente sunt definite prin ecuațiile (16.18). Există *k* este numărul din mulțimea {l,2.3| care este diferit de *i* și J.

(a) Pentru IS/.;S3

R<j - cos^cos^ $_7$ (l-cos0)-cos^ $_4$ sin0(-l) $^{<+/}$ *dacă i<J* (16.18a)

R $_{tJ}$ - cos^ $_($ cos^(l-cos0)+cos* $_4$ sin0(-l)' $^{+y}$ *dacă i>j* (16.18b)

«, - cos ^{2} 0 $_z$ (l-cos0)+cos0 (16.18c)

(b) Pentru / -4 sau; -4

^ $_4$ -0 IS/S3 (16.18d)

^4,-0 1S/S3 (16.18e)

«44-1 (16.180

Dacă axa de rotație coincide cu una dintre axele de coordonate, atunci expresiile de mai sus se simplifică considerabil și se reduc.

de fapt, la formula de rotație pe plan.

16.4 PROIECȚII ORTOGONALE

Proiecțiile sunt probabil cea mai importantă clasă de transformări în grafică, deoarece produc vederi bidimensionale ale obiectelor tridimensionale. Cel mai simplu tip de proiecție este proiecția *ortog onală* în care imaginea unui punct se găsește ca piciorul normalului de la punct la planul de proiecție. Dacă planul de proiecție este planul *xy* , atunci pur și simplu se stabilește *z* egal cu zero (sau o altă constantă).

Exemplul 16.2: Un cub este definit de opt vârfuri: /I(0,0,0), 5(2,0,0), C(2,2,0), D(0,2,0), £(0,0,2), £(2,0,2), G(2,2,2) și //(0,2,2). Găsiți proiecția ortogonală după ce este rotită cu 60’ în jurul axei care unește punctele (0,0,0) și (2,2,2).

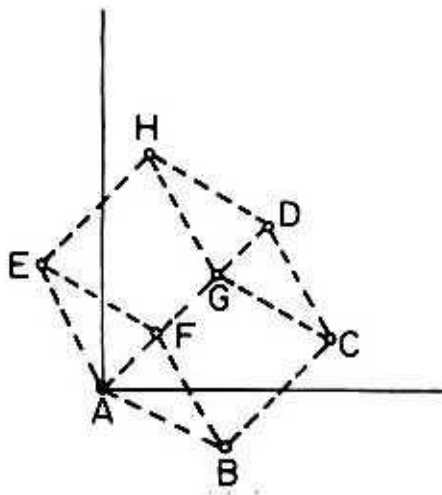


Figure 16.2 Location of the cube vertices used in Example 16.2

In this case cos^i — cos$\$_2$ — cos0$_3$ = 1/VJ and the first group of elements of the rotation matrix is given by

Rij '' j()-cos6) - J-.sin0(-l)$^{'+/}$ if f <J

^{R}*tj* '' y(l-cos^) + y=-sin0(-l)$^{f*'}$ if/>/

Ru = y(l —cos0) + cosfl

pentru orice fl. Pentru 5=60' avem următoarea matrice

2 2 $-\frac{1}{3}$
3 3
1 2 2
R - 3 (16.19)
3 2
2 3
$-\frac{1}{3}$ 3

Nu listăm ultima coloană și ultimul rând din *R* , deoarece acestea vor fi date de ecuația (16.18df) și, în acest exemplu, nu vom folosi coordonate omogene. Vârfurile cubului vor fi acum în următoarele locații:

, (0.0.0) . «,14.1,, 44,, 444. £(4±4). Hy.yT" G < 2 - 2 - 2 >- și ,,(yyf K

Proiecția la z = 0 va consta din următoarele opt puncte:

4(0,0), «(aa). c <y4>- °44>- £("f4"

2 8
G(2,2) și //(—,—). Acestea sunt prezentate în Figura 16.2. Pentru a

produce o proiecție adecvată, trebuie să decidem care dintre punctele ^T'T^ an ^ ^V'T^ W

*H ^ viiWe ' Th® are primul z — — y și

secunda z — —. Dacă privim cubul de sus, va fi vizibil cel cu z mai mare, rezultând

Figura 16.3a. Altfel, noi the will

have Figure 16.3b. □

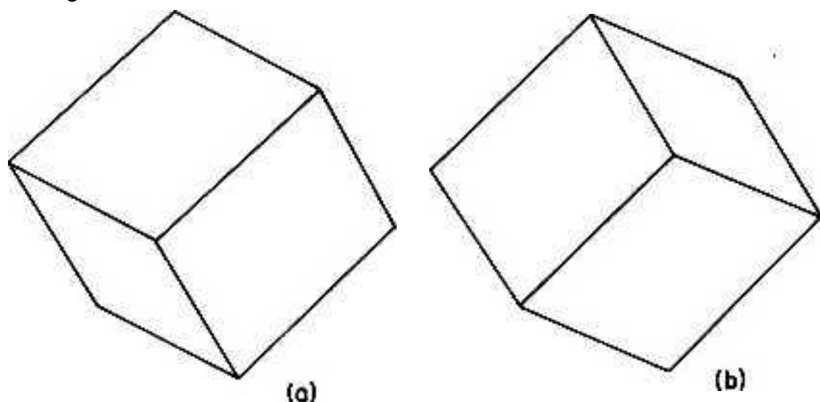


Figure 16.3 Possible views of the cube of Example 16.2

Deși proiecțiile ortogonale sunt ușor de obținut, nu o fac

oferiți scene realiste, cu excepția cazului în care se presupune că observatorii sunt foarte

departe de obiecte. Când distanța lor față de scenă este comparabilă cu dimensiunile acesteia, trebuie să folosim proiecții în perspectivă.

16.5 PROIECȚII DE PERSPECTIVĂ

Proiecția în perspectivă încearcă să imite modul în care un observator își formează o impresie asupra unei scene. Obiectele sunt proiectate pe un plan vizual dintr-un punct central, ochiul observatorului (Figura 16.4).

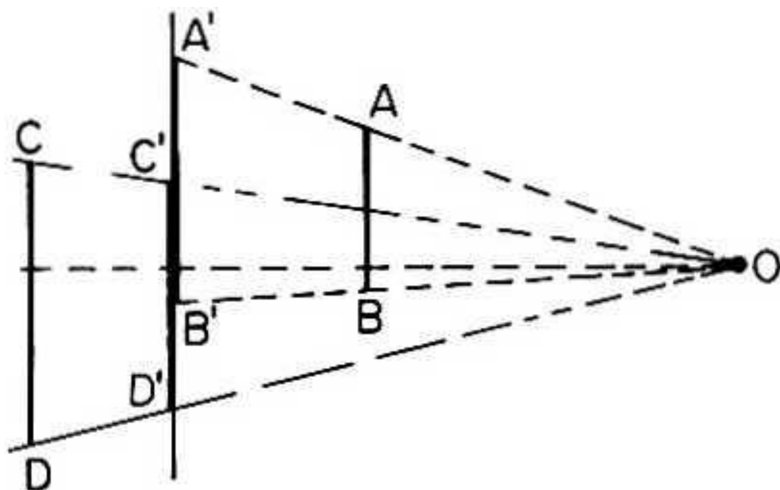
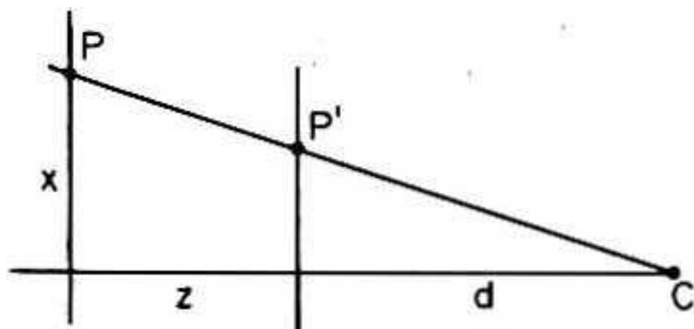


Figura 16.4 **Ilustrarea proiecțiilor în perspectivă. Segmentul** *AB* **este proiectat ca** *A'B',* **iar CD ca** *CD'*

O astfel de imagine proiectată poate fi construită cu ușurință din descrierea solidelor și a fost un instrument artistic de multe secole. Trebuie subliniat că foarte puțini artiști respectă strict regulile de perspectivă, deoarece imaginile create de reguli nu arată întotdeauna corect. Din păcate, în grafica computerizată, trebuie să urmați o regulă precisă, deoarece programele de calculator pot implementa doar formule matematice. Dacă nu suntem mulțumiți de rezultatele aplicării regulilor stricte ale perspectivei, trebuie să le modificăm într-un mod matematic explicit. Locația centrului de proiecție este de obicei considerată poziția unui vizualizator al ecranului la o distanță *d* de centru, care poate fi, de asemenea, Uken ca origine a coordonatelor. Folosind noțiunea din figura 16.5, aflăm că un punct *P* cu coordonatele (x^,z) va fi mapat în *P'* cu coordonate

-^- 0

d + *zd* + *z*



Figwre 16.5 Relația dintre coordonatele unui punct și cele ale proiecției sale în perspectivă

Pentru a afișa rezultatul, totuși, trebuie să luăm în considerare care obiecte le ascund pe altele și, prin urmare, este necesar să se păstreze informații despre coordonata z. Din motive care vor fi explicate mai târziu, alegem să aplicăm o scalare similară pe z ca pe x și *y.* Acest lucru nu afectează distanțele relative ale obiectelor față de ecran deoarece dacă Z| > z 2 > 0 este de asemenea adevărat că

dZt dz.

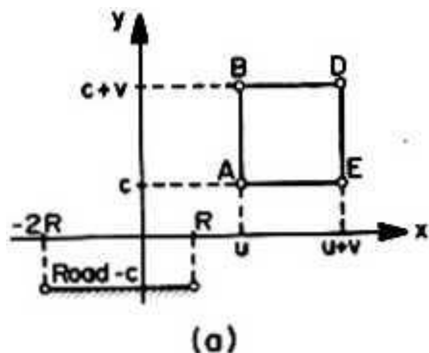
Transformarea perspectivei poate fi exprimată într-un mod elegant în coordonate omogene deoarece trebuie să modificăm doar ultima componentă. În special, putem scrie

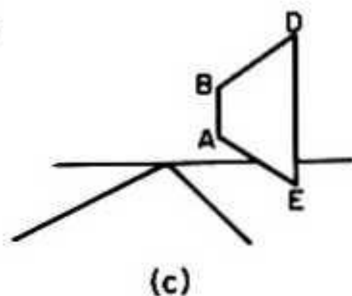
$$\mathbf{p} - \begin{vmatrix} 10\ 0\ 0 \\ 0\ 10\ 0 \\ 0\ 0\ 10 \\ 0\ 0\ 7\ 1 \end{vmatrix} \qquad (16,20)$$

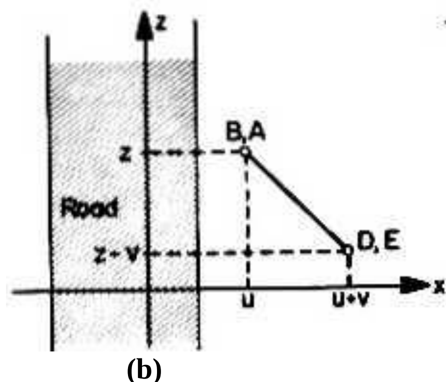
Dacă ($,*.£.*) sunt coordonatele omogene, constatăm că după proiecție

^.^i.,0,1).^)

de când ^*wz.* Aceasta oferă factorul de scalare corect. Pentru a afișa o scenă, trebuie mai întâi să înmulțim toate punctele cu P, apoi să decideți care sunt vizibile și, în final, să iluminați punctele ecranului ale căror coordonate *x,y* corespund cu cele ale obiectelor vizibile.







(b)

Flfure 16.6 Aranjarea obiectelor pentru Exemplul 16.3. (a) arată locația coordonatelor x și *y* în timp ce (b) arată x și *z.* (c) este o proiecție.

Exemplul 16.3: Creați un afișaj care simulează vederea unei persoane care conduce pe un drum drept. Așezăm observatorul la originea coordonatelor și presupunem că observatorul se mișcă pe o direcție perpendiculară pe ecran. Deoarece sistemul de coordonate se mișcă și în această direcție, putem crea diferite vederi prin schimbarea coordonatei *z* a obiectelor din scenă. Pentru simplitate, presupunem că, pe lângă drum și orizont, există un singur alt obiect în scenă, un panou publicitar pătrat situat așa cum se arată în figurile 16.6a și 16.6b. Formează un unghi de 45' cu axa drumului, partea sa cea mai stângă se află la distanța u de linia de-a lungul căreia se deplasează observatorul, are înălțimea v, lățime \/2v și distanța dintre colțul său inferior și observator.

este *c.* Atunci coordonatele colțurilor sale vor fi:

$$A = \begin{vmatrix} u \\ c \\ z \end{vmatrix} \quad u \quad 8 - c+v \quad D - \begin{matrix} u+v \\ c+v \\ Z\sim V \end{matrix} \quad E = \begin{matrix} u+v \\ c \\ z-v \end{matrix} \qquad (16.21)$$

Distanța dintre suprafața drumului și observator este *c,* marginea dreaptă este la o distanță /?, iar marginea stângă la o distanță 2/? de la observator. Prin urmare, cele două drepte paralele care delimitează drumul vor fi date de următoarele perechi de ecuații:

x - +^ > = -c (marginea dreaptă) (16.22a)

x - *-2R y* = *-c* (muchie din stânga) (16.22b)

Observăm că unele părți ale afișajului vor rămâne neschimbate. Un punct al muchiei drepte la o distanță *z* de observator va fi afișat la

$$x = \frac{Rd}{d + z} \quad y = -\frac{cd}{d + z} \qquad (16.23)$$

Pentru z—co găsim x = y — 0 și pentru z - 0, x — R, *Y ~ -c.* Acolo deci. marginea dreaptă va apărea întotdeauna ca un segment de linie de la punctul *{Rc)* până la punctul (0,0). În mod similar, linia din stânga va fi un punct care conectează punctele (-2R,-c) și (0,0). Acest lucru este prezentat în Figura 16.6c. Proiecțiile în perspectivă ale colțurilor panoului publicitar vor avea coordonate

ud

d+z

(16.24a)

i£±*M

d+z

l*d2K *(u+v)d*

d+z-v *d+z-v*

O'- (I6.24b)

(c+v)d

d+z—v *d+z—v*

Pentru z —co toate punctele se prăbușesc la origine. Dacă afișajul are o rezoluție de *N* pixeli, atunci cea mai mică valoare perceptibilă este *\/N,* astfel încât, dacă *c* < *u,* cea mai mare valoare a lui *z* care oferă coordonate diferite de zero va fi *d(Nc-\).* Programul va consta dintr-o buclă care începe de la z *=-d(Nc—l)* și coboară până la z — 0. Ar trebui să aveți grijă să tăiați afișajul deoarece punctele £ și *D* vor fi primele care vor dispărea din vedere. Dacă se poate face o selecție adecvată de biți în cuvintele memoriei de reîmprospătare (vezi secțiunile 1.6 și 1.7), atunci se poate șterge zona de pe
ecran și reafișează-l fără a afecta fundalul. Afișarea plăcii se poate face folosind un algoritm de umplere simplificat pentru întreaga zonă. □

Exemplul arată că proiecțiile în perspectivă sunt simple atunci când nu există interacțiuni între obiecte. Dacă *T* este matricea de mișcare dintre cadre și *P* matricea de proiecție, se procedează prin înmulțirea tuturor vectorilor (omogene) « cu PT, tăierea scenei în raport cu fereastra de vizualizare și, în final, afișarea rezultatelor. Problema devine mai dificilă atunci când obiectele interferează între ele și problema liniei ascunse trebuie rezolvată.

16.6 NOTE BIBLIOGRAFICE

Dovada pentru formulele de rotație în jurul unei axe se bazează pe [16.WI], desigur o referință destul de timpurie. Din cauza complexității sale, formula pare să fi dispărut în majoritatea tratatelor moderne și chiar cărțile de grafică par să o ignore (de exemplu [l.NS]). Chiar și atunci când este prezentată, dovada este dată de manipularea laborioasă pentru mula, mai degrabă decât de reprezentarea diadică elegantă a lui Wil son și Gibbs [16.WI]. Nu este surprinzător că geometria proiectivă oferă multe instrumente pentru studiul proiecțiilor. Cititorul trebuie să știe totuși că relația dintre un obiect și proiecția sa în perspectivă este numită pur și simplu *o perspectivă* în literatura respectivă [14.FI], (14.MA], [14.PE], [14.RO] și [14.TO] Rezultatele proiecțiilor în perspectivă sunt adesea contraintuitive (vezi problema 16.8), deoarece ele

distorsionează unele aplicații de perspectivă ingenioase. scene „imposibile”.

16.7 LITERATURA RELEVANTA

[16.ER] Ernst, B. *The Magic Mirror of MCEscher,* New York: Ballantine Books, 1976.

[16.PF] Pach, K. și Frey, T. *Vector and Tensor Analysis.* Budapesta: Terra, 1964.

[16SA] Salmon, G. *Geometria analitică a trei dimensiuni,* voi. I, ediția 7** , New York: Chelsea, 1927.

[16.WI] Wilson, EB *Vector Analysis,* (fondat pe prelegerile lui JW Gibbs), New Haven: Yale University Press, 1901. *(Fourth Printing. 1922.)*

16.8 PROBLEME

16.1. Demonstrați ecuația (16.2) și ecuația (16.3).

16.2. (a) Să se arate că aria unui triunghi cu vârfuri la (x^jz), (ˣ 2.J'2.z). iar (xjj'j.z) este dat de formula

aria « y [xi(j $_2$ ~ J's) + x $_2$ (y $_3$ -^i) + x $_3$ (/, -y $_2$)

(b) Găsiți expresia ariei când coordonatele z ale vârfurilor nu sunt al) aceleași.

16.3. Furnizați o dovadă detaliată a Propoziției 16.2.

16.4. Demonstrați cele patru părți ale Propoziției 16.3.

16.5. Găsiți matricea de rotație pentru o rotație în jurul axei y.

16.6. Găsiți transformarea pentru proiecția ortogonală pe planul definit de cele trei puncte (1,0,0), (0,1,0) și (0,0,1). *El:* Ar trebui să găsiți utilă expresia pentru matrice din ecuația (16.19).

16.7. Arătați că fiecare dintre cele patru puncte A_p, B_p, D_p și E_p din Exemplul 16.3 se deplasează de-a lungul unei linii care trece prin (0,0) în timp ce z se schimbă.

16.8. Arătați că proiecția ortogonală a unei sfere este întotdeauna un cerc, în timp ce proiecția în perspectivă poate fi fie un cerc, fie o elipsă. (Ultimul rezultat este contraintuitiv pentru majoritatea oamenilor. Este una dintre abaterile perspectivei artistice din perspectiva matematică. Următorul experiment vă poate convinge de adevărul său. Țineți o lanternă lângă o minge de baschet într-o încăpere întunecată. Umbra de pe perete va fi o elipsă, cu excepția cazului în care linia care unește sursa de lumină și centrul sferei este perpendiculară.)

16.9. Extindeți editorul de puncte introdus în Capitolul 10 pentru a permite manipularea construcțiilor tridimensionale. În acest scop, trebuie să introduceți capacitatea de transformări tridimensionale .

Capitolul 17

CREAREA TRIDIMENSIONAL AFIȘARE GRAFICE

17.1 INTRODUCERE

Folosim termenul de *afișare tridimensională* pentru afișajele care sunt proiecții ale scenelor tridimensionale, mai degrabă decât pentru afișajele în trei dimensiuni.! Cu toate acestea, utilizarea unor operații de proiecție adecvate, eliminarea suprafețelor care nu ar trebui să fie vizibile și umbrirea pot crea o impresie realistă de adâncime, așa cum se poate observa din figurile 17.1 (Plansa 28) și 17.2 (Plansa 29). Problemele de vizibilitate sunt centrale în crearea unor astfel de afișaje, la fel ca algoritmii de umbrire (Secțiunea 13.9) și tehnicile de randomizare pentru introducerea aspectului texturii, așa cum se arată în Figura 17.2. Soluția eficientă a problemelor de vizibilitate este probabil unul dintre pașii majori în crearea unui astfel de afișaj. Acestea sunt adesea denumite *linia ascunsă* sau problemele *de suprafață ascunsă .* Secțiunea 17.2 este o discuție generală a subiectului. Secțiunea 17.3 se ocupă de a t Afișajele cu adevărat tridimensionale pot fi create prin mijloace holografice, o tehnologie care se află în afara domeniului de aplicare al acestui text. Afișările stereografice necesită crearea a două vederi, fiecare dintre acestea putând fi generată cu tehnicile descrise în acest capitol, deși nu câștigăm să intrăm în metode de coordonare precisă între cele două. Vezi [I7JH1 pentru un exemplu.

algoritm de vizibilitate folosind un arbore quad și Secțiunea 17.4 cu un algoritm deosebit de potrivit pentru grafica raster. Secțiunea 17.5 discută subiectul *coerenței:* cum să profitați de interdependența diferitelor părți ale unei scene pentru a accelera soluționarea problemei de vizibilitate . Chiar dacă există un număr mare de algoritmi de vizibilitate în literatură, prezentăm aici doar doi dintre cei mai simpli. Motivul este că alegerea unei anumite metode depinde în mare măsură de aplicație. Este în primul rând o chestiune de a pune împreună unele dintre tehnicile pe care le-am discutat mai devreme, în special algoritmii de tăiere și umplere. Secțiunea 17.6 discută modificările necesare pentru aplicarea unor astfel de algoritmi la descrierile de obiecte neliniare. În cele din urmă, secțiunea 17.7 analizează pe scurt utilizarea randomizării și reflecțiilor multiple.

În general, problemele de vizibilitate pot fi atacate fie în spațiul obiectului, fie în spațiul imaginii. O soluție în spațiul obiectului se concentrează pe relațiile geometrice dintre părțile obiectelor pentru a decide ce este vizibil și ce nu. O soluție din spațiul imaginii traversează zona imaginii și examinează proiecțiile pe măsură ce apar. Ambii algoritmi descriși în Secțiunile 17.3 și 17.4 operează în spațiul imaginii. Secțiunea 17.5 discută un exemplu în spațiul obiect.

Pentru a afișa obiecte complexe, este necesar să existe descrieri matematice adecvate pentru acestea. Două soluții sunt utilizate în mod obișnuit. Într-una, un set de curbe este desenat pe un solid real (sec Figura 17.3a), iar descrierea matematică a acestor curbe este folosită pentru a genera afișajul. O astfel de metodă este deosebit de potrivită pentru grafica vectorială, dar afișajele rezultate au un cadru de sârmă .

A doua soluție necesită împărțirea suprafeței solidului într-un set de petice. O aproximare matematică simplă poate fi găsită peste fiecare petic, iar afișarea obiectului este produsă ca poarta agregată a afișajelor acestor petice de suprafață. Figura 17.3b prezintă un exemplu. Adesea, peticele de suprafață sunt triunghiuri plane, deoarece triunghiul unei suprafețe este un proces destul de simplu, deoarece trei puncte definesc un plan. De asemenea, afișarea triunghiurilor, umplerea lor și găsirea intersecțiilor pe perechi sunt simple. Dezavantajul major al acestei reprezentări este că s-ar putea să fie nevoie de un număr foarte mare de triunghiuri pentru a produce impresia unei suprafețe netede. Patch-urile de suprafață de tipurile discutate în Secțiunile 13.4 până la 13.7 oferă o alternativă și cele mai moderne afișaje le folosesc pentru descrierea obiectelor.

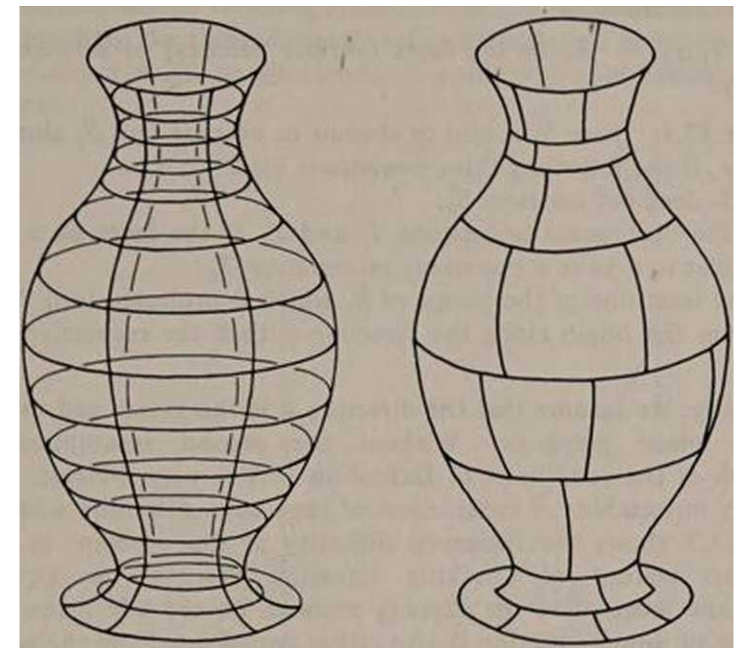


Figura 17.3 **(a) Reprezentarea unui obiect solid printr-un set de curbe, (b) Reprezentarea unui obiect solid printr-un set de pete de suprafață.**

17.2 LINIA ASCUNSĂ ȘI PROBLEME CU SUPRAFAȚĂ ASCUNSĂ

Afișarea cubului din Exemplul 16.2 ilustrează o instanță simplă a problemei liniei ascunse. Deși există douăsprezece linii pe obiect, doar nouă dintre ele ar trebui să fie afișate pentru a crea impresia unui afișaj solid. Problema poate fi exprimată și în termeni de suprafețe ascunse, luând în considerare fețele cubului. În grafica vectorială, se ocupă de obicei cu linii ascunse, în timp ce în graficele raster , problema este exprimată în termeni de suprafețe ascunse. Importanța problemei nu este din cauza dificultății sale inerente, ci din cauza cerinței ca soluția să fie calculată rapid, de exemplu, în timp ce un obiect este rotit. Primul pas în rezolvarea acestei probleme este o generalizare a materialului din Secțiunea 14.4.4 la trei dimensiuni, folosind rezultatele Secțiunii 16.2.

17.2.1 Umbra de suprafață

Fie 5,^2. - - * S M fie fețele (petice de suprafață) ale tuturor obiectelor de pe afișaj.

Definiția 17.1: Fața *Si* se spune că umbră sau ascunde fața *S }* de-a lungul direcției v, dacă sunt îndeplinite următoarele trei condiții:

(a) *Si* nu intersectează *S^*

(b) proiecțiile ortogonale 7} și *T }* ale fețelor pe un plan perpendicular pe v au o intersecție nevidă 7},;

(c) cel puțin unul dintre punctele lui *S t* care este proiectat în *Ty* se află mai departe de origine de-a lungul direcției v decât punctul respectiv al lui Sy. □

De obicei, presupunem că direcția v este axa z și că fețele sunt poligoane plane. Fără a doua ipoteză, verificarea condițiilor din Definiția 17.1 este foarte dificilă, dacă nu complet insolubilă. O comparație a definiției de mai sus cu poziția Pro 14.2 arată dificultatea crescută a problemei în trei dimensiuni: în loc de a verifica Ecuația (14.21a) sau Ecuația (I4.2lb) trebuie să rezolvăm problema de tăiere pe planul *xy* . Verificarea non-intersecției este, de asemenea, destul de complicată. În plan, un segment de linie este definit în mod unic de cele două puncte finale ale sale, iar analiza secțiunii 14.4.2 este direct aplicabilă. În spațiul tridimensional, o față nu trebuie să fie un triunghi, deci selectarea a trei puncte pentru verificarea semnelor determinanților din Propoziția 16.2 nu este suficientă. Trebuie să le verificați pentru toate vârfurile ambelor fețe. Locația față de origine poate fi verificată prin înlocuirea unuia dintre punctele unuia dintre planuri în ecuație cu celălalt, după selectarea unui sistem din dreapta pe acesta din urmă. Următoarea definiție este o modificare a Definiției 17.1 pentru proiecțiile în perspectivă.

Definiția 17.2: Fața *S {* se spune că umbră sau întunecă fața *Sj,* **când** este privită din punctul C, dacă sunt îndeplinite următoarele trei condiții:

(a) *Sj* nu intersectează *S t* \

(b) proiecțiile în perspectivă *T t* și 7) ale fețelor de pe un plan care nu conține *C* au o intersecție nevidă *T If ',*

(c) cel puțin unul dintre punctele lui *S,* care este proiectat în *T* , *se* află mai departe de-a lungul liniei de la *C* decât punctul respectiv al lui Sy. □

Este posibil să se evite distincția dintre proiecții dacă folosim expresiile din Secțiunea 16.5 pentru proiecțiile în perspectivă. Pentru a face asta,

construim scene rezultate din proiectii de perspectiva in doi pasi. Mai întâi aplicăm transformarea tridimensională dată de matricea *P* a ecuației (16.20) pe toate obiectele. Un punct cu coordonatele *x,y,z* se va mapa în

$$x' = \frac{dx}{z+d}, \quad y$$

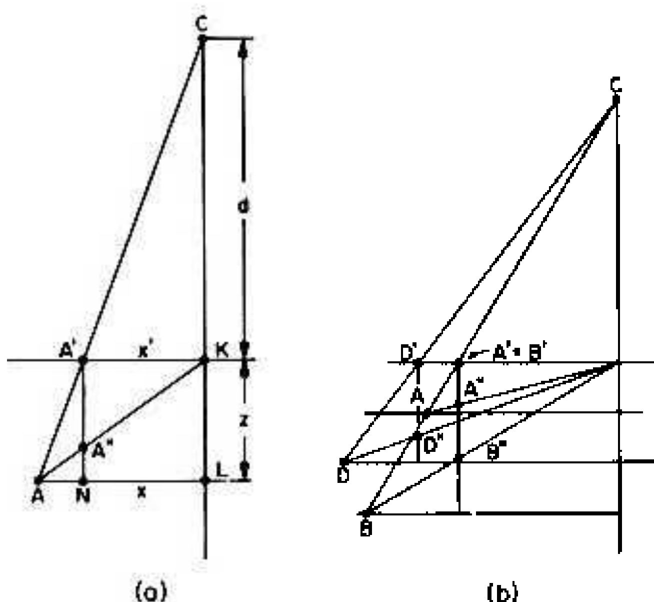


Figura 17.4 (a) Construcția geometrică a transformării dată de ecuația (16.20). *A"* este imaginea lui 4. (b) Aceeași transformare aplicată unui grup de puncte.

Figura 17.4a ilustrează o interpretare geometrică a acestor expresii pe planul *xz* . Conform geometriei elementare avem

$$\frac{A'K}{AL} \qquad \frac{CK}{CL} \qquad \frac{d}{d+z'}$$

întrucât *AL* = *x* și *A'K* — *x'*. Asemenea

$$\frac{A'A''}{UN} \qquad \frac{CK}{CL} \qquad \frac{d}{d+z'}$$

deoarece *A'N* = *KL* = *z* și *A'A"* = *z'.* Prin urmare *A"* este imaginea lui *A*

sub această transformare. Figura)7.4b prezintă transformarea aplicată unui grup de puncte. Putem vedea din ecuația (17.1) precum și din figura 17.4 că imaginile punctelor situate pe aceeași linie de la centrul de proiecție *C* (cum ar fi *A* și B) se aliniază de-a lungul unei linii verticale până la planul de proiecție (cum ar fi *A"* și *B").* Punctele care se află pe linii diferite de *C* (cum ar fi *A* și D) au imagini care nu se aliniază vertical (cum ar fi *A"* și *D").*

Acum putem continua cu al doilea pas. Verificați umbra suprafeței . nu cu obiectele originale, ci cu imaginile lor sub ecuația de transformare (16.20) și proiecții ortogonale. Rezumam rezultatele noastre după cum urmează:

Propoziţia 17.1: Rezolvarea unei probleme de vizibilitate sub proiecţiile de perspectivă este echivalentă cu rezolvarea unei probleme de vizibilitate pentru imaginile obiectelor sub transformarea ecuaţiei (16.20) şi proiecţiile ortogonale. □

17.2.2 Abordări ale problemei de vizibilitate

Definiția 17.1 sau 17.2 necesită să examinăm toate perechile de fețe pentru a găsi care dintre ele le ascund pe altele. Acest lucru nu este fezabil pentru afișajele care conțin mii de astfel de fețe, dar s-ar putea face pentru fețele unui singur obiect. Într-adevăr, să fie o scenă să conţină *k* obiecte şi să fie numărul de suprafeţe pentru obiectul /'* *s* $_t$. Atunci o comandă totală va necesita efort proporțional cu

A (17,2)

(-1 în timp ce compararea numai a suprafețelor din cadrul fiecărui obiect ar necesita un efort proporțional cu

S'? (17,3)
(„Eu

De obicei, obiectele nu se intersectează astfel încât se poate lua în considerare soluția problemei umbrei suprafeței pentru contururile lor, necesitând efort proporțional cu A^2. Această abordare necesită o contabilitate atentă. Mulți algoritmi de vizibilitate rezolvă dificultatea transformând problema tridimensională într-o problemă bidimensională într-unul din următoarele moduri:

(a) Luați în considerare proiecțiile poligonale ale fețelor și, folosind un algoritm de tăiere, găsiți toate perechile care se intersectează (inclusiv poligoane în care

unul îl înconjoară complet pe celălalt). Apoi aplicarea Propoziției 16.1 va arăta care dintre ele îl ascunde pe celălalt. În loc de a verifica perechile direct, se poate lua în considerare intersecția lor cu ferestre care sunt dreptunghiuri obișnuite. Două poligoane se pot întuneca unul pe celălalt numai dacă intersectează aceeași fereastră. Algoritmul 17.) se bazează pe această idee.

(b) Intersectați toate fețele cu un plan orizontal și rezolvați problema de vizibilitate pentru segmentele rezultate. Această abordare este implementată de algoritmul 17.2.

17.2.3 Vizibilitatea unui singur obiect convex

Dacă avem de-a face cu un singur obiect, iar acel obiect este convex, atunci putem rezolva problema vizibilității examinând direcțiile normalelor către fețele obiectului. Acestea pot fi găsite cu metodele din Secțiunea 13.2. Dacă norma) arată departe de ecran, atunci avem o față din spate care este invizibilă. Dacă normalul este orientat către ecran, fața este vizibilă. (Direcția normalei poate fi găsită cu ușurință din semnul lui *F,* definit în ecuația (13.5c).) A se vedea discuția din Secțiunea 17.5 despre coerența geometrică pentru unele tehnici înrudite.

17.3 ALGORITM DE VIZIBILITATE A ARBORELOR QUAD

Acest algoritm ia în considerare ferestrele afișajului care corespund nodurilor arborelui quad (vezi Secțiunea 6.4). Dacă nu există margini într-o fereastră, atunci culoarea acesteia poate fi decisă printr-o simplă ordonare în direcția *z* . Acesta este cazul ferestrelor 1), 22, 24, 42, 44 și 33 din Figura 17.5. (Face parte din fundal.) Dacă există margini, atunci fereastra este subdivizată coborând arborele quad. Figura 17.5 prezintă subdiviziunile pentru încă două niveluri după partiția inițială. Ferestrele omogene includ acum 1413, 143] etc. În cele din urmă, partiția continuă până la nivelul unui singur pixel. Marginile conturului pixelilor pot primi o valoare (luminozitate sau culoare) în funcție de cea a vecinilor lor. În mod clar, metoda seamănă foarte mult cu un proces de segmentare , folosind „nicio margine prezentă în regiune” ca predicat de uniformitate. (Din punct de vedere istoric. lucrurile au venit invers. Vezi Notele bibliografice.) Marginile care nu sunt vizibile vor dispărea automat deoarece regiunile de pe ambele părți ale acestora vor avea aceeași culoare, cea a suprafeței de obturare.

Efortul principal în acest algoritm are loc în a decide dacă o față intersectează o fereastră. Există trei posibilități: (a) fața înconjoară fereastra, (b) se intersectează sau se află în interiorul ferestrei sau (c) este complet disjuns de fereastră (Figura 17.6). Decizia poate

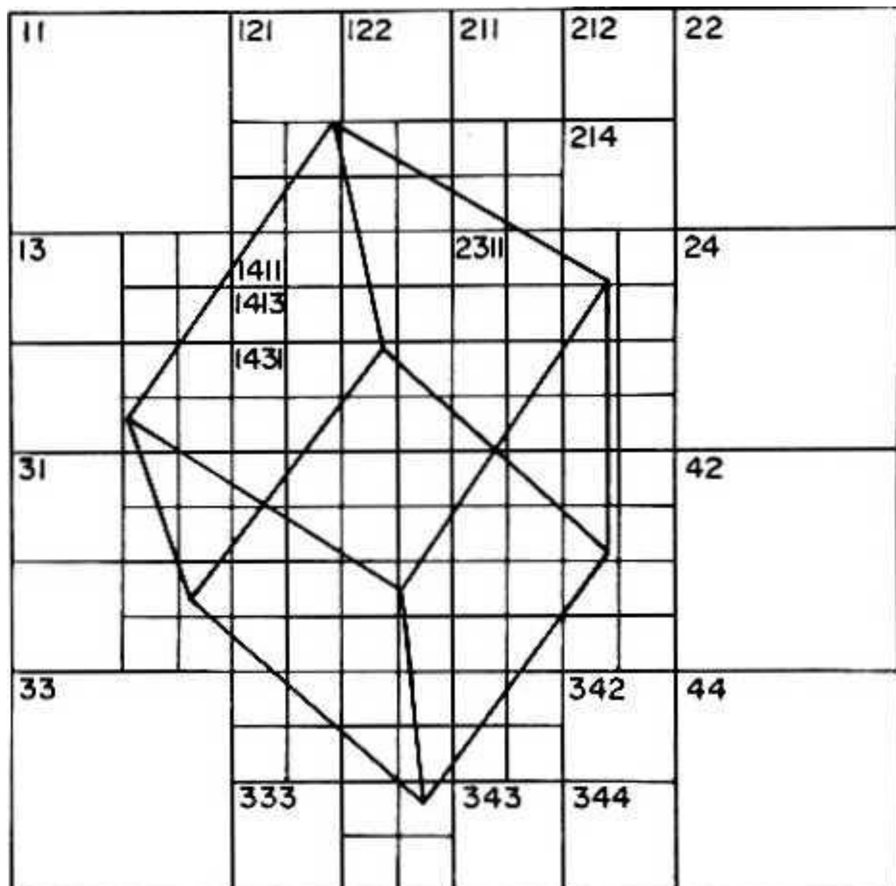


Figura 17.5 **Utilizarea unui algoritm de linie ascunsă arbore cvadru. Proiecțiile marginilor obiectului sunt prezentate în linii grele, iar liniile subțiri delimitează pătratele corespunzătoare primelor trei niveluri ale arborelui quad.**

se face prin rezolvarea problemei de tăiere pentru fața *F* și fereastra FT folosind algoritmul 15.2 pentru toate părțile feței. Procesul poate fi simplificat deoarece algoritmul nu are nevoie să cunoască intersecția reală a poligoanelor, doar dacă se intersectează sau nu (vezi problema 17.3). Să numim *LOC(WF)* o procedură care implementează o astfel de metodă și returnează următoarele valori: doi pentru cazul (a) când fața *F* înconjoară fereastra *W,* așa cum este ilustrat în Figura 17.6a; una pentru oricare dintre cele două configurații din cazul (b) când fața *F* este fie în interiorul *lui W* , fie intersectează *W* (Figura 17.6b); și zero pentru cazul (c) când *F* și *W* sunt disjunse (Figura 17.6c).

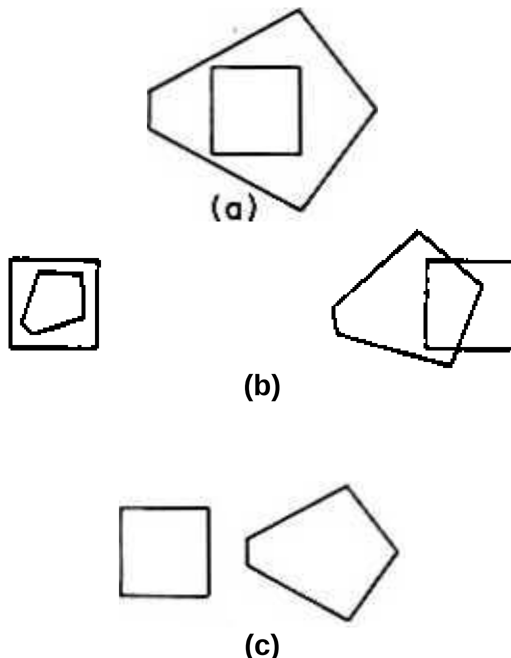


Figura 17.6 Relații posibile între o față și o fereastră. Valorile lui *K* **returnate de procedura** *LOC* **sunt două pentru (a), una pentru ambele cazuri prezentate în (b). și zero pentru (c).**

Algoritmul începe cu o partiție a zonei de afișare în ferestre corespunzătoare frunzelor unui arbore quad. Menține două liste pentru fiecare fereastră: *L* conține toate fețele din afișaj care pot intersecta nodul; iar *M* conține acele fețe care înconjoară nodul. La început *M* este gol, iar *L* are toate fețele; dar la sfârșit *L* este gol și *M* conține acele fețe care înconjoară nodul. Pentru fiecare fereastră algoritmul apelează procedura *LOC* pentru toate fețele din *L* (pasul 4 din Algoritmul 17.1). Inițial numărul de astfel de fețe va fi foarte mare, dar cele mai multe dintre ele vor fi eliminate deoarece sunt în afara ferestrei . Această eliminare se face în pasul 5. Dacă *LOC* returnează valoarea zero pentru toate fețele pentru o anumită fereastră, aceasta înseamnă că fereastra are culoarea fundalului și nu va mai fi examinată (steagul *F* este lăsat egal cu zero). Când valoarea returnată este două. fata este transferata de *la L la M* (pasul 7). Dacă zero sau două sunt singurele valori returnate pentru o fereastră, această fereastră poate fi, de asemenea, pusă deoparte. Este înconjurat complet de una sau mai multe fețe și culoarea sa va fi cea a feței cea mai apropiată de observator. Acesta poate fi găsit căutând în

Algoritmul 17.1 Algoritmul cu linii ascunse de arbore cvadru

Notație: 0 este un pătrat corespunzător unei frunze a arborelui quad. Pentru fiecare astfel de pătrat algoritmul menține două liste. £ și A/. £ este lista tuturor

fețelor care pot intersecta *Q, M* lista fețelor care au fost găsite pentru a înconjura *Q.* Steagul *F* este setat la unul pentru toate pătratele în care vizibilitatea nu a fost încă decisă.

0. Selectați un nivel de pornire al arborelui quad și pentru fiecare pătrat corespunzător unui nod creați o listă £ care să conțină toate fețele solidelor din afișaj. Lista *M* va fi goală pentru fiecare pătrat. Inițializați matricea de flag F la 1.

1. **Pentru** fiecare pătrat *Q* unde *F(Q)* este egal cu una, faceți:

ÎNCEPE.

2. Setați *F(Q)* la zero.

3. **Pentru** fiecare față a afișajului *P* din *L{Q)* **faceți:**

ÎNCEPE.

4. Apelați *LOC(Q y P)* și fie *K* valoarea returnată.

5. **Dacă** *K* este egal cu 0, **atunci** îndepărtați *P* din *L(Q)* {fața *P* minciuni în afara pătratului *Q}.*

6. **Dacă** A este egal cu 1, atunci înlocuiți *Q* cu copiii săi în quad tree și puneți-le să moștenească listele £ și *M* din *Q* {fața *P* intersectează pătratul *Q],* setați *F* la unul pentru fiecare dintre copii și rupeți din buclă.

7. **Dacă** *K* este egal cu 2. **atunci plasați** *P* **în** *M(Q)* **și îndepărtați** -
l

din *L(Q).* {fața *P* înconjoară pătratul *Q}.*

Sfârşit.

Sfârşit.

8. **Pentru fiecare pătrat** *Q* **faceți:**

ÎNCEPE.

Dacă *M* este gol, **atunci** afișați culoarea fundalului.

10. În caz contrar, căutați în conținutul lui *M* fața cea mai apropiată de observator și afișați-i culoarea.

Sfârşit.

11. **Sfârșitul algoritmului.**

conținutul lui *M* la pasul 8. Dacă valoarea returnată este una pentru o anumită față, atunci trebuie să subdivizăm fereastra (pasul 6). În mod clar, orice fețe care au fost disjunse din fereastra originală vor fi, de asemenea, disjunse de copiii săi în arborele quad. De asemenea, fețele care înconjoară fereastra îi vor înconjura și copiii. Deoarece o subdiviziune are loc prima dată când *LOC* returnează valoarea 1, știm că orice modificări în listele £ și *M* s-au datorat unor fețe disjunse sau care conțin. Prin urmare, listele pot fi
moştenit de copiii ferestrei.t Faţa care provoacă întoarcerea lui 1 trebuie să rămână în £ pentru toţi copiii.

Consultați Secțiunea 17.5 pentru câteva modificări posibile ale pasului de inițializare. De asemenea, este posibilă accelerarea operației prin modificarea pasului 4, astfel încât o față care intersectează un pătrat să fie comparată cu elementele lui A/.

Dacă se află în spatele unuia dintre ei, este ignorat.

17.4 UN ALGORITM DE VIZIBILITATE DE SCANARE A LINII RASTER

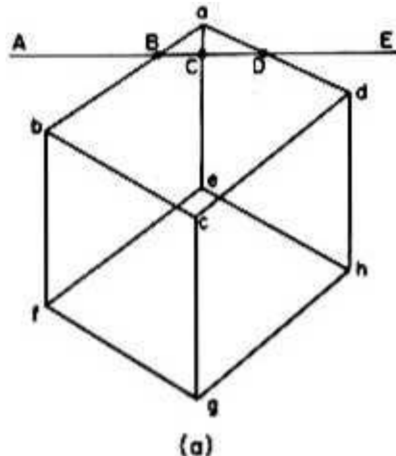
Un algoritm de scanare a liniilor raster rezolvă problema vizibilității în timp ce generează afișarea linie cu linie. Dacă toate obiectele sunt regiuni plane și dacă nu există suprapuneri între ele, algoritmul se reduce la umplerea marginilor din Algoritmul 8.1. (Secțiunea 8.2.) În acest caz, marginile contururilor de umplut au fost sortate în direcția *y* . **Aici luăm în considerare intersecțiile obiectelor cu plane a căror ecuație este j» — constantă. Pentru obiectele poliedrice, astfel de intersecții vor fi poligoane în planul *xz* , iar segmentele de linie care trebuie umplute sunt proiecții ale acestor poligoane pe axa x.

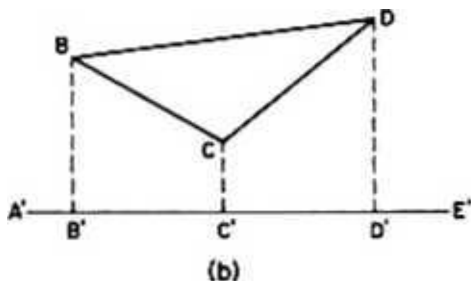
Figura)7.7a prezintă un cub (proiectat pe planul *xy*) și a *y* = plan constant (proiectat ca o linie) care intersectează muchiile cubului *ab* la *B, ae* la *C* și *ad* la *D.* Figura 17.7b este o proiecție a intersecției pe planul *xz* , arătând și proiecția ecranului de-a lungul ver țurilor *A'E* [f] *.* Acestea sunt punctele *B', C'* și *D'.* Pentru acel exemplu, problema vizibilității se reduce la a decide dacă algoritmul de umplere va umple intervalul *B'D'* cu culoarea feței *abed* sau, în schimb, va umple intervalele *B'C* și *CD'* cu culoarea fețelor *abfe* și respectiv *aehd* .

Algoritmul 17.2 implementează aceste idei. Partea principală este formată din pașii de la 3 la 6, iar concizia listării se datorează faptului că unii dintre pași sunt exprimați în termeni de operațiuni descrise în detaliu în alte părți ale textului. Dacă scena de afișat variază în timp, atunci trebuie găsite pentru fiecare *y* intersecțiile tuturor fețelor cu planuri *y* — constantă. Acest lucru se face în pasul 4. Necesită aplicarea unui algoritm de tăiere, cum ar fi algoritmul 15.1 (sau algoritmul 15.3 dacă fețele nu sunt convexe). Poligonul este proiecția feței pe planul *xy* , iar linia este intersecția planului constant ay cu planul de proiecție. Rezultatul va fi un set de segmente de dreaptă pe planul *xy* .

t Ar trebui să fim atenți în implementările algoritmului să avem listele copiate mai degrabă decât partajate, deoarece listele pot fi modificate ulterior în moduri diferite pentru fiecare dintre copiii nodului.

**Presupunem că *z* este coordonata perpendiculară pe ecran și *y* coordonatele perpendiculare pe direcția de scanare de pe ecran.





Flprt 17.7 Funcționarea algoritmului de vizibilitate a liniei «can, (a) proiecția unui cub de afișat și ay- plan constant pe planul *x~y* , (b) proiecția intersecției pe planul *xz* și de-a lungul unei linii de scanare.

Pe de altă parte, dacă aceeași scenă urmează să fie afișată în mod repetat, atunci putem îmbunătăți eficiența algoritmului oferind descrierea obiectelor în termeni de poligoane care sunt intersecțiile lor cu *y* — planuri constante. În plus, astfel de intersecții pot fi sortate în funcție de valoarea lui *y.* (Acest lucru este similar cu stocarea contururilor sortate în funcție de valoarea *y* din algoritmul 8.1.) Apoi pasul 4 va parcurge pur și simplu lista de astfel de poligoane.

După pasul 4 trebuie să luăm în considerare care segmente umbră pe altele, așa cum sa discutat în Secțiunea 14.4.4. S-ar putea alege să aplice această abordare

Algoritmul 17.2 Linia de scanare Algoritmul suprafeței ascunse

Notație: **Array *COLOR* este un buffer în care este stocată culoarea unui pixel care urmează să fie afișat, Array *DEPTH* conține valoarea lui *z* pentru punctul cel mai apropiat de observator la fiecare x al liniei de scanare. *P* este o intersecție a unei fețe cu linia de scanare *R.* un segment de linie. x_p și z, sunt coordonatele punctelor *p* ale lui *P.***

1. **Pentru** fiecare linie de scanare *R* a ecranului faceți:

ÎNCEPE.

2. Inițializați matricea *COLOR* la culoarea de fundal și matricea *DEPTH* la cel mai mare *z posibil.*

3. **Pentru** fiecare față *F* a obiectelor scenei **faceți: Begia.**

4. **Dacă** *F* intersectează planul orizontal cu ecuația *y — R,* **atunci** găsiți intersecția *P* și **faceți:**

ÎNCEPE.

5. **Pentru** fiecare punct *p* al intersecției *P* **faceți:**

ÎNCEPE.

6. **Dacă** *z,* este mai mic decât D[x,J. **apoi** setați C[x_p] egal cu culoarea lui *P* și setați *D[*x_p*]* egal cu [10] *• **End.**

Sfârşit.

Sfârşit.

7. Afișează *CULOARE.*

Sfârşit.

8. **Sfârșitul algoritmului.**

direct și să vină cu o listă de intervale care sunt vizibile. Algoritmul 17.2 urmează o cale diferită. Menține două matrice *COLOR* și *DEPTH* ale căror intrări corespund pixelilor de-a lungul liniei de scanare. *COLOR* conține culoarea unui anumit pixel și *DEPTH* coordonata *z a segmentului cel* mai apropiat de observator. Pentru fiecare linie de scanare, *COLOR* este inițializată la culoarea de fundal și *DEPTH* la cea mai mare - valoare posibilă a lui *z.* Când algoritmul examinează un segment în planul *xz* (de exemplu, *BC* în Figura 17.7) el compară valorile *z* ale punctelor iu cu valoarea în *PROFINȚĂ* pentru același x. Dacă un punct are o valoare de *z* mai mică decât cea din *DEPTH.* apoi valoarea în *DEPTH* este actualizată și culoarea punctului intră în punctul respectiv al *COLOR.* În mod clar, după ce toate segmentele au fost examinate, *COLOR* conține o copie corectă a scenei și poate fi afișată.

17.5 COERENȚĂ

Algoritmul 17.1 și algoritmul 17.2 sunt ineficienti deoarece nu fac nicio presupunere cu privire la dimensiunea sau pozițiile relative ale fețelor. Ele sunt aplicabile unei scene care conține un număr de fețe poligonale în poziții aleatorii. Acesta este rareori cazul în care există multe relații între diferitele părți ale imaginii. Astfel de interdependențe sunt denumite *coerență.* Vom discuta pe scurt despre modul în care algoritmii pot fi îmbunătățiți, listând în același timp diferitele tipuri de coerență utilizate în grafică. Coerența poate fi specificată în mai multe moduri și poate fi exprimată fie în spațiul obiectului, fie în spațiul imaginii. Următoarea este o listă de posibilități:

Coerența marginilor: vizibilitatea unei margini se schimbă numai atunci când traversează o altă margine. Se poate profita de această proprietate prin crearea unei liste de segmente de margine fără intersecții. Apoi trebuie să verificăm doar un punct din fiecare segment. Găsirea intersecției implică o anumită cantitate de muncă, astfel încât să se poată profita de proprietate doar pentru afișări repetate ale aceleiași scene.

Coerența feței: Din cauza dimensiunii mici a fețelor poliedrice în comparație cu imaginea totală, este adesea adevărat că dacă o parte a feței este vizibilă, întreaga față este vizibilă.

Coerența obiectului: Vizibilitatea unui obiect poate fi adesea decisă prin examinarea unui solid circumscris, care poate avea o formă simplă: o sferă sau un poliedru cu fețe paralele cu planurile de coordonate. Aceasta este o extensie a ideii utilizate pentru găsirea intersecției poligoanelor din Secțiunea 15.6 și poate fi aplicată în mai multe moduri. De exemplu, pasul 4 al algoritmului 17.2 poate fi simplificat prin menținerea intervalului de valori > (maximum și minim) pentru fiecare obiect într-o listă separată. Sau putem folosi coerența obiectului în algoritmul 17.1 pentru a reduce dimensiunea inițială a listei *L.* În cele din urmă, poate face detectarea relațiilor de umbră la suprafață mai ușoară, așa cum sa discutat în Secțiunea 17.2.2.

Coerența liniei de scanare: segmentele care sunt vizibile într-o linie sunt, de asemenea, probabil să fie vizibile în linia următoare. Dacă avem o estimare a dimensiunii minime a obiectului, atunci putem alege un număr *K* și putem modifica algoritmul 17.2 după cum urmează. După ce examinăm și afișam o linie de scanare , *sărim* K — 1 linii și examinăm următoarea linie. *1,^.* Dacă aceleași segmente sunt vizibile pe ambele linii, fie *F* setul de fețe unde le aparțin. Apoi, pentru liniile sărite, calculăm matricea *C* folosind numai fețele din *F* și afișăm liniile *K* împreună. (Putem evita cu totul rezolvarea problemei de vizibilitate pentru liniile intermediare prin interpolarea partițiilor dintre cele ale lui $I_{(}$ și $I^{\wedge}{}_{K}$*.)* Dacă segmentele vizibile în I_{t+K} sunt diferite de cele vizibile în $I_{lt,}$ atunci putem reveni la o linie intermediară, să spunem *1^^/^* și să repetăm același lucru

analiză. Procesul poate fi organizat într-o manieră similară cu algoritmul de potrivire a poligoanelor discutat în Secțiunea 12.5. Această abordare va da rezultate corecte dacă niciun obiect nu este suficient de mic pentru a fi cuprins în întregime în *K* linii de scanare.

Coerența cadrului: imaginea nu se schimbă prea mult de la cadru la cadru. Pentru a utiliza acest tip de coerență în mod eficient, trebuie să păstrați o listă de obiecte în mișcare și de fundal. Apoi algoritmul de afișare trebuie să reexamineze doar zona din apropierea obiectelor în mișcare de la cadru la cadru. Amintiți-vă și discuția din Secțiunea 1.6 despre tratarea obiectelor în mișcare.

Coerență geometrică: numărul posibilelor configurații vizibile/invizibile pe orice vârf este limitat.

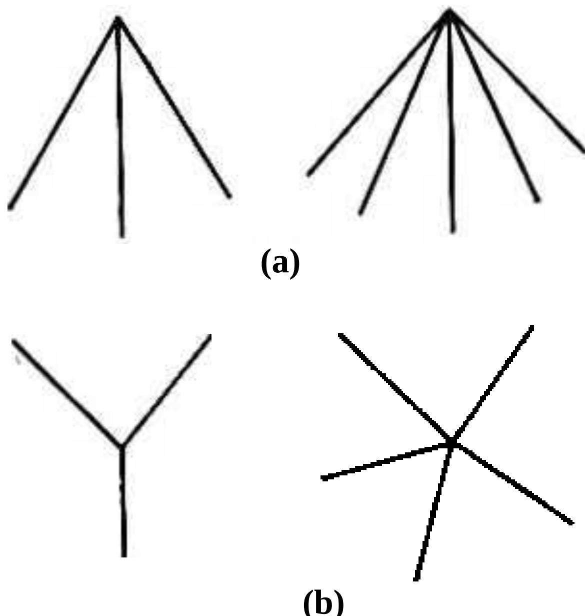


Figura 17.8 (a) Dacă muchiile prezentate aparțin unui obiect convex, atunci cele două extreme trebuie să fie vizibile, (b) Pentru un obiect convex, toate muchiile care înconjoară un vârf au aceeași vizibilitate.

Ultimul caz este unul care a primit puțină atenție în grafică, dar a fost discutat pe scară largă în recunoașterea modelelor picturale (analiza scenei). Conduce la reguli foarte simple dacă știm că toate solidele sunt convexe. În special, dacă unghiul maxim dintre muchiile convergente este sub 180' (Figura 17.8a), atunci cele două margini extreme sunt vizibile, iar vizibilitatea celorlalte muchii depinde de locația lor relativă.

la planul definit de primele două. Dacă unghiul depășește 180’, atunci toate marginile au aceeași vizibilitate (Figura 17.8b). Cel mai bun mod de a folosi coerența geometrică este de a rezolva mai întâi problema de vizibilitate pentru fiecare obiect și de a-i eticheta marginile sau fețele în consecință. Când afișajul este generat , trebuie să vă preocupați

doar de fețele vizibile ale fiecărui obiect și acest lucru reduce numărul de articole care trebuie sortate. Dacă vârfurile sau marginile conturului proiecției *(ABCGHE* în Figura 17.9) primesc o etichetă specială și obiectul este convex, atunci este ușor de stabilit coerența obiectului sau cadrului. Pentru exemplul dat să presupunem că am determinat că *D* este vizibil în primul cadru. Atâta timp cât *D* rămâne în poligonul *ABCGHE,* nu va exista nicio modificare a vizibilității fețelor. Consultați Notele Bibliografice pentru unele dintre cele mai recente abordări ale problemei.

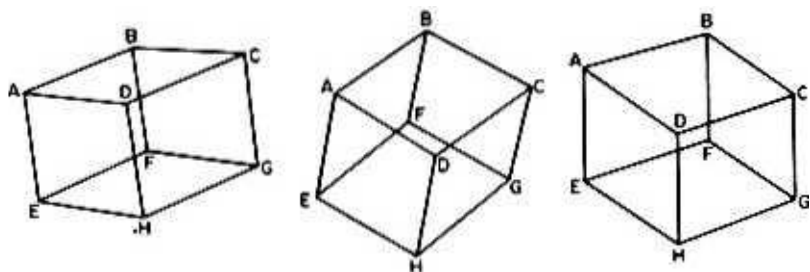
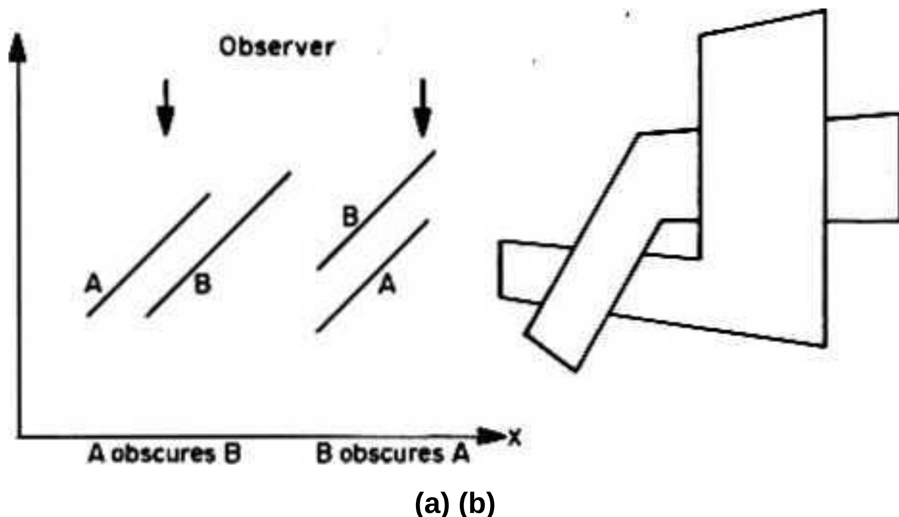


Figura 17.9 Stabilirea coerenței cadrului obiectului. Fețele *ABCD, ADHE* și *DHGC* au aceeași vizibilitate în toate cele trei cadre.

Ar trebui să subliniem că multe tehnici simple de rezolvare a problemelor de vizibilitate eșuează în situații netriviale. Figura 17.10a arată că extremele în direcția *2* nu sunt suficiente. Dacă P este un punct de intersecție a proiecțiilor celor două segmente, atunci vizibilitatea poate fi decisă prin compararea numai a punctelor PA și P_B *care* se proiectează în *P.* Această tehnică este valabilă și atunci când ambele fețe sunt regiuni plane convexe. Dacă suprafețele sunt regiuni plane, dar nu convexe, atunci este necesară o examinare mai detaliată pentru a lua în considerare cazurile de tipul prezentat în Figura 17.10b. Calculele pentru suprafețe neplanare sunt încă mai extinse decât calculele pentru plane. Subdiviziunea peticelor de suprafață în petice mai mici poate simplifica problema.



(a) (b)

Figura 17.10 **(a)** Exemplu în care extremele în direcția z nu sunt suficiente pentru a determina vizibilitatea. Pe de altă parte , vizibilitatea poate fi determinată prin compararea punctelor *PA* și P_a. (b) Exemplu de două suprafețe plane care se ascund reciproc.

17.6 DESCRIERI OBIECTE NELINIARE

Algoritmul 17.2 este destul de general și nu depinde deloc de forma suprafețelor utilizate pentru fețe, atâta timp cât se ignoră detaliile calculului. Algoritmul necesită efectuarea următoarelor operații: (a) Aflați dacă un plan *y* = constantă intersectează o față, (b) Dacă planul intersectează fața, atunci găsiți intersecția . (c) Căutați cea mai apropiată curbă în direcția *z* . Aceste operații sunt foarte simple când fețele sunt poligoane plane convexe. Într-adevăr, pentru (a) trebuie doar să verificăm valorile extreme ale lui *y* pentru limita feței. Operațiunile (b) și (c) se referă numai la segmente de linie . Dacă afișajul trebuie să fie umbrit, este nevoie și de normala la suprafață la fiecare segment (vezi Secțiunea 13.9). Pentru un poligon plan normala este fixă pentru toate punctele. Dacă folosim patch-uri neliniare, pierdem aceste simplificări. Pentru a afla dacă un petec intersectează un plan, trebuie să luăm în considerare nu numai granița petei, ci și *silueta* proiecției sale din cauza posibilității unei umflături. Figura 17.11 prezintă două astfel de siluete pentru un petic sferic. Deoarece normala la suprafață nu mai este fixată, toate punctele interioare trebuie examinate .

(a) (b)

Flj«'« 17-H Silueta# de petice de suprafaţă sferică

Fie S(w,y) ecuația parametrică a suprafeței și presupunem că curbele S(uO). S(ul), S(0,v) și S(l,v) au legat plasturele. O astfel de reprezentare este posibilă pentru suprafețele Coons (Secțiunea 13.6), suprafețele Bezier (Secțiunea 13.7.1). sau suprafețe B-spUne (Secțiunea 13.7.2). Reamintim că S(uO) este o curbă dată pentru suprafețele Coon, așa cum se arată în ecuația (13.18'). și un polinom Bezier (vezi ecuația 13.31) sau un B-spline (vezi ecuația 13.33) pentru celelalte două cazuri. Același lucru este valabil și pentru celelalte curbe de delimitare. Fie ^(«.v), /(«,*) și Z(u,v) coordonatele punctului S(uv). Pentru a găsi intersecția cu un anumit plan > - c trebuie să rezolvăm ecuația

$$r(u,v)-c \qquad (17,4)$$

În general, o astfel de ecuație impune o dependență între u și v și dacă v(u) este o expresie a lui v în termeni de *u,* atunci S(u,v(u)) este o curbă pe planul *y ^ c.* O astfel de abordare generală nu este practică și, în schimb, se poate încerca să găsească o soluție numerică pentru fiecare dintre următoarele patru ecuații

$$r(u,o) -c \qquad (17.5a)$$

$$r(u,i)-e \qquad (17.5b)$$

$$r(o,v)-c \qquad (17.5c)$$

$$Hl,v)-c \qquad (17,5d)$$

Rezultatele vor specifica cele patru puncte în care limitele petec intersectează planul *y — c.* Dacă nu există soluții pentru aceste ecuații , aceasta nu înseamnă că planul nu intersectează petecul, deoarece poate face acest lucru în interior. Silueta suprafeței care este proiectată pe un plan perpendicular pe axa z este formată din acele puncte în care planul tangent la suprafață este paralel cu axa z. Prin

urmare, normala la suprafață nu trebuie să aibă nicio componentă paralelă cu axa respectivă. Conform ecuației (13.5c), astfel de puncte trebuie să satisfacă ecuația

$$X..Y,-Y_u X,-0 \quad (17,6)$$

unde indicele denotă derivate parțiale. În plus, punctele trebuie să satisfacă ecuația (17.4). Dacă vrem să găsim intervalul de valori *y* ocupat de patch trebuie să găsim extremele în direcția y. Acestea sunt fie puncte de limită, fie locuri în care variația lui S(u,v) în direcția lor este zero, și anume

$$-\hat{}(_w,v)-0, -\hat{}(«,v)-0 \quad (17,7)$$

Pentru umbrire trebuie să găsim coordonatele exacte pentru un anumit x, adică trebuie să rezolvăm un sistem format din ecuația (17.6) și

$$*(u,v)-x \quad (17,8)$$

Toate aceste ecuații sunt neliniare și sunt de obicei rezolvate prin tehnici numerice care procedează prin ghicirea unei soluții și apoi corectând-o prin iterații.! Într-un astfel de caz, coerența liniei de scanare devine foarte importantă deoarece soluțiile dintr-o linie pot fi folosite ca ipoteze pentru soluțiile din linia următoare. Soluțiile analitice sunt posibile doar în cazuri foarte simple. Vă prezentăm unul dintre ele ca exemplu pentru a ilustra conceptele relevante, deși nu și metodele de soluționare.

Exemplul 17.1: Considerăm cazul unui sector sferic dat de următoarele ecuații

$$X(u,v) - \cos u \cos v , K(u,v) — \sin v , Z(u,v) — \sin u \cos v , \quad (17.9a)$$

şi

$$x+ziO. \quad (17.9b)$$

În formă neparametrică Ecuația (17.9a) este echivalentă cu

$$x^2+y^2+z^2-1 . \quad (17\hat{})$$

t Această abordare a fost propusă de Blinn IIIBU, care a folosit metoda lui Newton pentru găsirea soluției. A se vedea (I7.LA) pentru un rezumat.

Proiecția pe planul xz are forma prezentată în figura 17.10a și constă din puncte care satisfac ecuația (17.9b) precum și inegalitatea

$$x^2 + z^2 < l. \qquad (1710)$$

Găsirea proiecției pe planul *xy* este mai dificilă. În acest caz, marginea patch-ului nu este descrisă de valorile lui *u* și v date mai sus, ci este definită ca intersecția sferei și a planului.

$$x+z-0. \qquad (17.10')$$

Eliminând *z* între această ecuație și ecuația (17.9a'), aflăm că proiecția muchiei este elipsa

$$2x^2 +/=l. \qquad (17.11)$$

Dacă lucrăm în formă parametrică, observăm că

x + *z* ™ V5sin(u + -y-jeosv .

Acesta va fi zero dacă fie v — — sau dacă u ------- -----—. Prima valoare cedează
2 4

x = z = 0 și *y* = 1

iar al doilea

$$x = \frac{1}{V2}7{=}rcosv\ ,\ z = \frac{1}{V2}\ \hat{}cosv\ ,\ y = sinv$$

care este ecuația parametrică a unei elipse în trei dimensiuni. Proiecția pe planul x-^ se găsește ignorând valoarea lui z. Cititorul poate verifica că aceasta este într-adevăr aceeași cu ecuația (17.11). Pentru orice valoare fixă a lui y găsim o pereche de valori ale lui x de forma

$$\pm\sqrt{\frac{1-y^2}{2}}$$

Cu toate acestea, doar unul dintre ele este un delimitator al proiecției. Pentru a examina silueta, calculăm expresia ecuației (17.7) care dă valoarea — sinucos ^{2} v. Aceasta devine zero fie pentru u = 0, fie pentru v = x/2. Din forma neparametrică putem găsi că z = 0. Silueta este atunci cercul

$$x^2 + y^2 = 1$$

iar proiecția completă constă dintr-un arc eliptic și unul circular, așa cum se arată în Figura 17.10b. □

O alternativă la soluția ecuațiilor neliniare este să

subdivizați peticele până când devin suficient de mici pentru a fi aproximate prin poligoane. Această abordare se bazează pe teorema 16.1, care afirmă că domeniul unui polinom Bezier poate fi împărțit în două, iar fiecare submulțime a curbei este din nou un polinom Bezier. Rezultate similare pot fi găsite pentru alte curbe.

17.7 FĂRĂ UN AFIȘARE NATURAL

Textura, randomizarea și reflexiile multiple ale luminii au fost toate folosite pentru a da afișajelor aspectul unei scene naturale. Un alt set de eforturi, îndreptate către același scop, s-a ocupat de eliminarea aspectului „scării" sau „zimțate" a reconstrucțiilor constante pe bucăți (vezi Secțiunea 2.4). Vom trece în revistă pe scurt tehnicile care au fost folosite pentru a produce impresia de textură.

După cum am discutat în Secțiunea 3.3, textura corespunde statisticilor de ordinul doi ale distribuției luminozității pe o imagine. Pentru a produce o astfel de impresie într-o imagine sintetizată trebuie să atribuim culoarea sau luminozitatea unui pixel pe baza culorii sau luminozității vecinilor săi. Există două moduri de a face acest lucru. Într-una presupunem că geometria suprafeței rămâne fixă, dată de S(u,v), dar culoarea se schimbă în funcție de o funcție C(u,v). Valorile lui *C(u,v)* pot fi date printr-o formulă sau sub formă tabelară, dar se pot genera și printr-un proces aleatoriu. O abordare mai realistă este de a permite variații ale suprafeței, cum ar fi ridurile și denivelările, și de a produce efectul texturii prin modificări ale luminii reflectate. O modalitate de a simula o astfel de rugozitate a suprafeței este de a perturba suprafața de-a lungul direcției normalei sale printr-o funcție (scalară) dată *P(u,v).* Dacă X^p, Y^p, Z^p indică valorile perturbate pe care le avem

'(u,v) -(u,v) + P(u,v) |jy|| (17.12a)

^(u,*) = Y(M,V) + ^(wv)-^- (17.12b)

Z'(M ,v)=Z(M ,v) + P(u,v)y^ (17.12c)

unde *(F_x,F_y,F_z)* sunt componentele normalei la suprafață date de ecuațiile (13.5) și ||F|| este norma ei. Pentru a calcula luminozitatea aparentă trebuie să găsim valorile normalei la suprafața sub perturbație. Fie X^p derivata parțială a lui X^p în raport cu *u.* Vom folosi notații similare pentru celelalte derivate . Apoi găsim că

F x d

X ∧ X " + ∧W + P ∧

W (l713)

Wc presupunem acum că *P(u,v)* ia valori foarte mici, dar că are derivate mari. Atunci este posibil să neglijăm cel de-al treilea termen al laturii de mână a ecuației (17.13). Wc poate găsi expresii similare pentru *Y^. Z'* etc. și astfel găsiți ecuația pentru normala perturbată să fie

Efectuând înmulțirile și folosind ecuația (13.5a) găsim după unele anulări

*F!-F x +(P y Z, -Z y P r)-^- + (Y y P ¥ -P y YJ-^* (17.15)

Dacă repetăm calculele pentru celelalte două componente ale normalului, găsim expresii similare și concluzionăm că normala perturbată poate fi găsită din normala neperturbată prin înmulțire cu o matrice simetrică 3X3. Matricea are unele pe diagonala sa principală și forma celorlalte elemente ale sale poate fi derivată din ecuația (17.15).

Funcția P(uv) poate fi dată fie în formă tabelară, fie în mod explicit, sau poate fi aleasă ca variabilă aleatorie. Consultați [13.BL] pentru mai multe despre acest subiect.

Dacă perturbarea suprafeței sau culoarea acesteia într-un punct trebuie să fie independentă de cea din alte puncte, atunci problema este foarte simplă. Tot ce trebuie să faceți este să apelați un (pseudo) generator de numere aleatoare în fiecare punct. Astfel de perturbații independente nu sunt un model realist al suprafețelor texturate fizice. Introducerea corelației de la punct la punct nu este prea complicată dacă cineva este dispus să formeze un tabel mare al funcției perturbatoare *P(u,v).* Totuși, nu se poate face „din mers", deoarece punctele care sunt văzute aproape în timpul creării afișajului pot fi foarte îndepărtate la suprafață. Consultați Notele Bibliografice pentru mai multe despre acest subiect și despre tehnicile conexe.

17.8 NOTE BIBLIOGRAFICE

Un studiu timpuriu al algoritmilor de vizibilitate poate fi găsit în [17SSS]. Algoritmul quad arbore enumerat aici se bazează pe unul dezvoltat de War nock (vezi (l.NS) sau [17.SSS] pentru o referință completă) și este prima utilizare publicată a acestei structuri de date în procesarea imaginilor. Algoritmul liniei de scanare se bazează pe unul de Watkins (vezi [l.NS] sau [17.SSS]). O colecție de metode mai recente poate fi găsită în [17.LA], care

include rezolvarea problemelor de vizibilitate cu pete de suprafață neliniare. [17.WH] prezintă o combinație de probleme de vizibilitate și umbrire , inclusiv reflexii multiple. [13.BL] oferă o analiză cuprinzătoare a unui algoritm de scanare raster atunci când obiectele sunt descrise prin patch-uri de suprafață. Include, de asemenea, o discuție extinsă a metodelor de producere a unui aspect texturat, iar Secțiunile 17.6 și 17.7 se bazează în mare măsură pe această lucrare. (Un rezumat al acestei teze poate fi găsit în [17.LA].) [17.FF] descrie o tehnică de randomizare care a fost utilizată cu succes pentru a produce imagini ale terenului.

[17.KC] descrie un algoritm de vizibilitate în spațiul obiectului când obiectele sunt sfere sau cilindri. Folosește o funcție similară cu *LOC* din Secțiunea 17.3, dar în loc să compare fețele și ferestrele compară fețele între ele. [17.MA] este o extensie a acestei lucrări cu introducerea umbririi (inclusiv a luminii). Se evită unele dintre dificultățile peticelor neliniare prin avansarea geometriei simple a sferei. Tehnicile descrise în aceste lucrări au fost folosite pentru a produce imagini ale modelelor moleculare. O altă utilizare a sferelor în grafică este descrisă în [17.BOT). Un model al unui obiect fizic este produs nu prin petice de suprafață, ci ca înveliș de sfere care se intersectează. Autorii prezintă un model al corpului uman format din aproximativ 300 de sfere și susțin că descrieri comparabile ar necesita până la 3000 de petice de suprafață. [17.ON] descrie lucrările conexe și [17.KN] prezintă elaborări ulterioare ale tehnicii .

Studiile recente ale problemei vizibilității au pus accent pe ordinea corectă a comparației feței și ferestrelor. [17.CL] este o discuție teoretică a abordării spațiului obiect. [17.FR] prezintă un algoritm de același tip ca și Algoritmul 17.1, dar cu o strategie atentă pentru examinarea suprapunerilor ferestrelor și fețelor. Rezultatul este o complexitate temporală liniară mai degrabă decât pătratică. [17.HZ] se ocupă de o combinație de coerență a cadrului și obiectului.

[17.HA) este o colecție de lucrări despre percepția pe care oamenii care proiectează ecrane o pot găsi utile.

17.9 LITERATURA RELEVANTA

[I7.BOT] Badler, NI; O'Rourke, J.; și Toltzis, H. „A Spherical Representation of a Human Body for Visualizing Movement". *IEEE Proceed ings* **67 (octombrie 1979), p. 1397-1403.**

[I7.CL] Clark. JH „Modele geometrice ierarhice pentru algoritmi de suprafață vizibilă". *CACM.* **19 (1976), pp. 547-554.**

(17.FF) Fournier, A. și Fussel. D. „Modelul stocastic" în graficul computerizat*." *CACM* **(în presă)**

II7.FR] **Franklin, WR „A Linear Exact Hidden Surface Algorithm”,** *SIG- GRAPH80.* **Seattle, Wash.. iulie 1980. pp. 117-123.**

II7.HA1 **Hagen, M. (ed.)** *The Perception of Pictures.* **New York: Academic Press, 1980.**

[I7.HZ] **Hubschman, H. și Zucker, SW „Coerența cadru-la-cadru și calculul suprafeței ascunse: constrângeri pentru o lume convexă”,** *SlGGRAPH'8l.* **Dallas, Texas. august 1981, p. 45-54.**

I17JH] **desJardins. M. şi Hasler. AF „Stereoscopic Displays of Atmospheric Model Dau”** *S/GGRAPH'80,* **Seattle. Wash.. iulie 1980, p. 134-146.**

[17.KC1 **Knowlton, K. și Cherry, L. „Atomi — A Three-D Opaque Molecular System for Color Pictures of Space-Filling or Ball-and- Stick Models”,** *Computers d Chemistry.* **1 (1977), p. 161-166.**

[I7.KNI **Knowlton, K. „Definirea asistată de calculator, manipularea și descrierea obiectelor compuse din sfere”,** *ACM Computer Graphics.* **IS (1981), p. 48-71.**

(I7.LA) **Lane, JM; Carpenter, LC; Whittcd, T.; și Blinn, JF „Scan Line Methods for Displaying Parametrically Defined Surfaces”,** *CACM.* **23 (1980), p. 23-34.**

II7.MA] **Max, NL „ATOMLLL - ATOMS with Shading and Highlights”** *SIGGRAPH'79,* **Chicago, august 1979, pp. 165-173.**

|I7.ON| **O'Rourke, J. și Badler, N. „Descompunerea obiectelor tridimensionale în sfere,”** */EEE Trans, on Pattern Analysis and Machine Intelligence.* **PAMI-I (197 9), p. 295-305.**

I17SSS1 **Sutherland, JE; Sproul. RF; și Schumacker, RA „A Char acterization of Ten Hidden-Surface Algorithms”,** *ACM Computing Surveys.* **6 (martie 1974), p. 1-56.**

[17.WH1 **Whittcd. T. „An Improved Illumination Model for Shaded Display”** *CACM.* **23 (1980), p. 34 3-349.**

17.10 PROBLEME

17.1. Investigați forma solidului care este imaginea sferei sub transformarea ecuației (16.20).

17.2. Scrieți un program care implementează algoritmul 17.1. Pentru a simplifica soluția problemei de tăiere presupunem că toate fețele sunt dreptunghiuri regulate paralele cu planul xy.

17.3. Implementați procedura *LOC(WF)* descrisă în Secțiunea 17.3. Rețineți că puteți omite acele părți ale algoritmilor de tăiere care calculează intersecția liniilor și, de asemenea, scurtați verificările, revenind la procedura de apel imediat ce este găsită o pereche de laturi care se intersectează. Pe de altă parte, o oarecare grijă este

necesare pentru a distinge cazurile când *F* înconjoară *W, W* înconjoară *F* și *F*

și W sunt reciproc *disjunctive* .

17.4. Implementați algoritmul 17.2 în condițiile menționate în problema 17.2.

17.5. Să facem un film! Pentru cea mai simplă versiune vom folosi două obiecte, un cub și un tetraedru obișnuit. Acesta din urmă ar trebui să aibă laturile care sunt aproximativ jumătate din lungimea laturilor cubului. Cubul ar trebui să fie poziționat cu una dintre diagonalele sale verticală și ar trebui să se rotească în jurul lui. (Alegerea atentă a unui sistem de coordonate va simplifica foarte mult formulele pentru transformarea rotației.) De asemenea, tetraedrul ar trebui să se rotească în jurul aceleiași axe la o distanță de aproximativ trei ori latura cubului. Viteza sa unghiulară ar trebui să fie de aproximativ o zecime din cea a cubului. Scrieți un program pentru a calcula o secvență de vederi ale scenei. Întregul aranjament ar trebui să semene cu cel al unei planete care se învârte în jurul Soarelui. Proiectul poate fi extins prin adăugarea mai multor obiecte, precum și prin creșterea numărului de fețe ale poliedrelor.

Sugestii: Acesta este un exemplu în care are sens să folosiți coerența obiectului pentru rezolvarea problemei de vizibilitate.! Majoritatea pixelilor vor aparține fundalului și de cele mai multe ori proiecțiile lor vor fi disjunse.

17.6. Repetați problema anterioară prin înlocuirea poliedrelor cu sfere. Deoarece acest lucru simplifică rezolvarea problemei de vizibilitate , introduceți următoarele complicații. Adăugați mai multe planete ale căror planuri de direcție sunt aproape normale cu planul de afișare, presupuneți că obiectul suționar este o sursă de lumină și umbriți în mod corespunzător suprafețele celorlalte obiecte.

f Când această temă a fost dată studenților de la Princeton, aproximativ jumătate dintre ei au folosit coerența obiectului, iar ceilalți au implementat un algoritm general în spațiul imaginii. Au avut la dispoziție aproximativ trei săptămâni pentru a finaliza proiectul.

INDEX AUTOR

Index refers only to full citations. Entries with the prefix P- refer to plates.

INDEX SUBIECTULUI

numerele aldine se referă la paginile cu definiția termenului. Numărul *uaiic* « se referă la paginile în care termenul este folosit într-o problemă, ff după un număr indică referire la nota de subsol.

partiția de cuvinte 13, 16
înfășurați în jurul valorii de 337

INDEX ALGORITMULUI

001.55
P338

32.95

001.55
P338

Pavlidis

AUTHOR

Algorithms for graphics and

TITLE

image processing

32.95

DATE LOANED | BORROWER S NAME | DATE RETURNED

Pavlidis *ffrfiPiCS* ^^ .Algoritmi pentru^procesarea imaginii



Biblioteca Carl S. Swisher
Colegiul Bethune-Cookman

DESPRE CARTE

Această carte oferă un tratament cuprinzător al procesării informațiilor picturale de către computer, un subiect care cuprinde grafica computerizată, procesarea imaginilor pe computer și recunoașterea modelelor picturale. Oferă instrumentele matematice necesare pentru cele trei domenii și conține o listă detaliată a algoritmilor. Exemplele și problemele de-a lungul întregii fac cartea potrivită ca text sau pentru auto-studiu.

DESPRE AUTOR

Theo Pavlidis a primit Diploma de Inginerie Mecanica si Electrica de la Universitatea Nationala Tehnica din Atena. Grecia, în 1957. iar MS și Ph.D. diplome în Inginerie Electrică de la Universitatea din California din Berkeley, în 1962 și, respectiv, 1964. El este la Centrul de Cercetare în Informatică al laboratoarelor Bell. Murray Hill. NJ. Între 1964 și 1980 a fost la departamentul de Inginerie Electrică și Informatică a Universității Princeton. În acea perioadă a fost și consultant pentru RCA. diverse laboratoare ale armatei americane și alte industrii În timpul anului universitar 1978-79 a fost profesor invitat la Universitatea din California din Berkeley. Interesele sale de cercetare se concentrează pe procesarea computerizată a informațiilor totale și includ recunoașterea modelelor, procesarea imaginilor și grafică. Este autorul a trei cărți și a peste o sută de lucrări tehnice. El este redactor principal al IEEE Transactions on Pattern Analysis and Machine Intelligence (PAM1) și editor asociat al unui număr de alte reviste. A fost președintele general al celei de-a cincea conferințe internaționale privind recunoașterea modelelor (decembrie 1980). Este membru al IEEE și membru al ACM. și Sigma-Xi.

HIBIIWH»* 22

• Cel mai bine este să evitați termenul de *editor grafic* , deoarece poate avea două semnificații: unul ca editor pentru a introduce date picturale și altul ca editor de text folosind un afișaj grafic. tf Din acest motiv, un astfel de editor este ideal ca proiect de laborator în derulare. Sarcinile de laborator sunt esențiale pentru studiul graficii și procesării imaginilor și se poate face o muncă considerabilă cu mijloace simple. De exemplu. Capitolul 5 descrie un proiect destul de complex folosind doar o imprimantă de linie ca dispozitiv de ieșire. O mare parte a efortului este

destinată procedurilor de intrare și ieșire, iar integrarea unor astfel de proiecte într-un editor are sens.

• Această notație este deosebit de semnificativă atunci când avem de-a face cu imagini cu două niveluri Acolo ne interesează doar pixelii întunecați (cu valoarea unu), care sunt șterși în timpul traversării

*i *2